VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Identifikace uživatelů webu bez použití cookies

Cookie-less Web User Identification

2020 Mikuláš Mascautanu

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Mikuláš Mascautanu**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Identifikace uživatelů webu, bez využití cookies
Cookie-less Web User Identification

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je analyzovat možnosti identifikace uživatelů webu bez využití souborů cookies a následně implementovat software, který tuto identifikaci umožňuje. Očekává se implementace klientské aplikace pro sběr dat o uživateli (JavaScript), tak serverové aplikace pro analýzu a vyhodnocení těchto dat (ASP.NET).

Hlavní body zadání:
1. Analýza technologií, které lze využít pro identifikaci uživatele webu, bez využití souborů cookies.
2. Návrh a implementace JavaScriptové knihovny pro sběr dat o uživateli.
3. Návrh a implementace serverové aplikace pro analýzu získaných dat o uživatelích.
4. Testování výsledného řešení napříč internetovými prohlížeči – analýza schopnosti detekce/rozeznání uživatelů.

Seznam doporučené odborné literatury:

[1] FLANAGAN, David. JavaScript: the definitive guide. 6th ed. Sebastopol, CA: O'Reilly, 2011. ISBN 978-0596805524.
[2] Pro Asp.net core MVC 2. New York, NY: Springer Science+Business Media, 2017. ISBN 978-1484231494.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Janoušek**

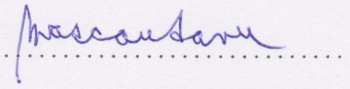
Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2020



Poděkování za pomoc při tvorbě této práce patří mnoha lidem. Především bych chtěl poděkovat Ing. Janu Janouškovi za poskytnutí důležitých rad a odborný dohled při tvorbě práce. Dále bych chtěl poděkovat všem důležitým lidem v mém okolí, kteří mi vždy nabídli pomoc.

Abstrakt

Cookies již nejsou jediným způsobem jak sledovat aktivity uživatelů internetu. V dnešní době existuje mnoho jiných řešení, jejichž praktičnost a spolehlivost se ovšem různí. Cílem této práce je vytvoření vlastní webové aplikace pro sběr dat o uživatelích, a také pro samotné identifikování uživatelů. Začátek je věnován vývoji cookies a analýze alternativních metod sledování. Dále se práce zabývá návrhem a implementací dvou nejzajímavějších metod vybraných na základě analýzy. Řešení se otestují napříč současně nejpoužívanějšími prohlížeči a mezi skutečnými uživateli. Výsledná úspěšnost metod se zhodnotí a navrhnou se možná opatření pro obranu proti nim.

Klíčová slova: web, identifikace prohlížeče, anonymita na internetu, fingerprinting, otisk prohlížeče, etag, mezipaměť, cookies

Abstract

Cookies are no longer the only way of tracking activities of internet users. Nowadays, there are plenty of different options some of which are more practical and reliable than others. The aim of this thesis is a developoment of a web application which collects data about users and identifies them. In the beginning, evolution of cookies and analysis of alternative tracking techniques are covered. Furthermore, the work focuses on designing and implementing two most remarkable methods which are picked based on the analysis. Both solutions will be tested across today's most used web browsers and among real users. Results of success rate will be evaluated and possible ways of protection proposed.

Key Words: web, browser identification, internet anonymity, fingerprinting, browser fingerprint, etag, cache, cookies

Obsah

Seznam použitých zkratek a symbolů

WWW	–	World Wide Web
HTTP	–	Hypertext Transfer Protocol
HTML	–	Hyper Text Markup Language
IETF	–	Internet Engineering Task Force
NPAPI	–	Netscape Plugin Application Programming Interface
LSO	–	Local Shared Object
MD5	–	Message-Digest algorithm
SHA	–	Secure Hash Algorithm
JSON	–	JavaScript Object Notation
GUI	–	Graphical User Interface
API	–	Application Programming Interface
DOM	–	Document Object Model
AJAX	–	Asynchronous Javascript And XML

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Anonymita na internetu je tématem diskuzí posledních let. Webové stránky se snaží sbírat informace o aktivitách svých návštěvníků, aby jim na jejich základě mohli vytvořit uživatelský profil pro marketingové účely v podobě cílené reklamy. V dnešní době se v tomto ohledu ovšem nelze spoléhat pouze na soubory cookies, protože o jejich využívání velká část veřejnosti ví a mohou se tak bránit.

Bakalářská práce se v první kapitole zabývá vývojem cookies. V další kapitole analyzuje alternativní metody sledování uživatelů internetu. Na základě této analýzy jsou vybrány dvě metody jevící se jako nejzajímavější, a tedy vhodné pro implementaci ve výsledné aplikaci: tzv. fingerprinting a využití mezipaměti prohlížečů. Fingerprinting je poměrně komplexní, a proto je mu věnována velká část práce. Získává různé běžně poskytované informace od prohlížečů a na základě srovnávání jejich podobnosti pak zpětně identifikuje prohlížeč daného uživatele. Největší potenciál této metody tkví v tom, že neukládá žádný identifikátor přímo do prohlížeče, a tím pádem uživatel svůj identifikátor nemá jak smazat. Na druhou stranu je metoda nepřesná a závislá na dostatečném množství unikátních informací mezi prohlížeči. Její použití za účelem cílené reklamy je ovšem dostačující, jelikož při omylu dojde pouze k zobrazení reklam cílených na někoho jiného.

Návrh a implementace obou vybraných metod ve vlastní aplikaci popisuje kapitola 3 a jejich úspěšnost skutečně identifikovat uživatele se testuje v kapitole 4. V případě fingerprintingu to kromě sběru dat o uživatelích zahrnuje také algoritmus na samotné identifikování podle posbíraných dat. Sběr dat proběhne napříč současně nejpoužívanějšími prohlížeči a mezi skutečnými uživateli a výsledky budou rozebrány do detailů. Poslední část práce popisuje některé způsoby, kterými se uživatelé mohou proti sledování bránit, a také zmiňuje nová opatření ze strany prohlížečů.

2 Vývoj cookies

Počátky popularizace WWW začaly v roce 1993 s příchodem prohlížeče Mosaic, který byl zdarma, multiplatformní a také jako první prohlížeč vůbec zobrazoval text a obrázky zároveň v jednom okně. Prohlížeče nepodporovaly tzv. relace (anglicky sessions), což z hlediska dnes tak často jmenované ochrany osobních údajů na internetu znamenalo větší soukromí a bezpečí. Internetový protokol HTTP, sloužící ke komunikaci mezi klientem (uživatelem) a webovým serverem, funguje způsobem dotazování a odpovídání, server si nepamatuje, jestli dotazu ze strany klienta předcházel jiný dotaz. Kvůli této vlastnosti se o protokolu říká, že je bezstavový [1]. Pokud uživatel navštívil webovou stránku, zavřel prohlížeč a vrátil se na ni zpět, jevil se webové stránce jako nový uživatel.

2.1 Počátky cookies

Již první veřejná verze prohlížeče Netscape Navigator vydaná v roce 1994 v sobě měla naimplementovanou podporu uchovávání relace v podobě cookies, o kterou se postaral Lou Montulli. Zmínku o cookies popsal v dokumentu Persistent Client State HTTP Cookies na webu Netscape Communications Corporation [2]. Tato "specifikacace" však byla velmi krátká, a ne příliš podrobná, a také nejspíš proto ji Netscape později smazal ze svého webu a najít se dá již jen na archivních stránkách.

V této době začínala tzv. válka prohlížečů mezi tehdy nejpopulárnějším Netscape Navigator a prohlížečem Internet Explorer, který byl teprve na vzestupu [3]. Tento prohlížeč od společnosti Microsoft implementoval podporu cookies již ve verzi 2.0 [4]. Výše zmiňované neformální popsání specifikace cookies od Lou Montulliho bylo převzato a několikrát předefinováno v publikovaných RFC (Request for Comments) dokumentech, které jsou dnes většinou vydávané velmi známou organizací IETF[1]. Poprvé se tak stalo z počátku roku 1997 v RFC 2109 [5]. Publikace, oproti původnímu dokumentu od Lou Montulliho, popisovala cookies a dva nové headery velmi podrobně, a navíc upravovala některé jejich vlastnosti. Dokument následně v roce 2000 překonal RFC 2965, který přidal tři nové hlavičky: Cookie, Cookie2 a Set-Cookie2 [6] a nakonec vydání RFC 6265 až v roce 2011, které hlavičky Cookie2 a Set-Cookie2 již neschvalovalo. Dodnes je tato publikace nejaktuálnější verzí RFC týkající se cookies a je považována za RFC standard [7].

2.2 HTTP protokol a cookies

Cookies dělají z původně bezstavového HTTP protokolu protokol stavový. Jsou jeho součástí, takže zařizují lepší zapouzdření před uživateli na rozdíl od tehdejších různých alternativ, představujících pouze dočasné řešení zachování relace mezi serverem a klientem. Server může v HTTP

[1] Komise pro technickou stránku internetu. Jde o organizaci produkující internetové standardy, která je tvořena dobrovolníky a zaměřuje se především na internetové protokoly.

odpovědi zaslat cookie v Set-Cookie hlavičce, aby zachoval stav relace. Další následující HTTP požadavky od klienta obsahují Cookie hlavičku. Může se jednat o jakoukoli informaci, kterou server vyžaduje, a to nejen pro již vícekrát zmiňované udržení relace s klientem. Webový server může tyto informace, neboli cookies, využívat také pro ukládání uživatelských nastavení nebo analýzu chování uživatele. Cookies jsou ukládány ve formátu textového řetězce, odděleny středníky a ukládány na straně klienta nebo serveru. U prvního případu ukládání je nevýhodou potenciální nedostatek hardwarového místa pro vyžadovaná data. Pro maximální kompatibilitu webové stránky mezi webovými prohlížeči se doporučuje maximálně 50 cookies s jednotlivou maximální velikostí 4093 bytů, ale u různých webových prohlížečů se tato maxima liší. V počítači uživatele bývá většinou uložen pouze jeho vygenerovaný identifikátor, podle něhož se mohou na straně serveru vyhledat všechny ostatní potřebné informace o uživateli. Tato varianta se často nazývá *sessions* a přináší výhodu v možnosti uchovávání teoreticky neomezeného množství dat, jelikož všechna data jsou uložena na serveru a v prohlížeči je uložen pouze identifikátor uživatele, pod kterým se data na serveru vyhledají [8]. Tento způsob párování identifikátoru uživatele na straně klienta a serveru se velmi často používá v různých metodách sledování uživatelů webu. Také je dobré zmínit, že při používání prohlížeče v režimu anonymního prohlížení se cookies vůbec neukládají.

2.3 JavaScript jako posun k dynamičnosti webu

JavaScript bude v průběhu práce poměrně často zmiňován, a proto je vhodné jej čtenáři představit více detailněji.

Brendan Eich v roce 1995 na požádání společnosti Netscape vytvořil programovací jazyk, který je dnes znám pod jménem JavaScript. O dva roky později, v roce 1997, byl jazyk standardizován organizací Ecma International a vydán jako specifikace skriptovacího jazyku pro webové stránky pod oficiálním názvem EcmaScript [9]. Použití JavaScriptu ve webových stránkách z něj dělá klientský programovací jazyk, což znamená, že jeho kód je spouštěn ve webovém prohlížeči, na straně klienta. Díky tomu umožňuje jazyk dynamicky měnit webovou stránku v prohlížeči uživatele, bez nutnosti ji aktualizovat. JavaScript má k dispozici (kromě dalších objektů) objekt `document`, představující DOM, ve kterém se nachází celá struktura HTML[2] stránky. DOM lze definovat jako platformově a jazykově nezávislé rozhraní umožňující dynamicky přistupovat k obsahu webového dokumentu [10]. Tento objekt má mnoho vlastností a jednou z nich je *document.cookie*, skrz kterou lze v JavaScriptu číst, ukládat a mazat cookies. Existuje zde jisté omezení, pokud má cookie nastavený příznak *HttpOnly*, JavaScript k této cookie nemůže přistupovat přes *document.cookie* [11].

Jednou podstatnou novinkou, kterou JavaScript později přinesl, je JSON. Jedná se o formát dat, ve kterém jsou data ukládána a čtena. JSON byl sice odvozen z JavaScriptu, ale je naprosto nezávislý na programovacím jazyku. Funguje jako dvojice klíč a hodnota, kde hodnota může být

[2]Značkovací jazyk (anglicky Hypertext Markup Language) sloužící primárně pro tvoření webových stránek

pouze objekt, pole, číslo, textový řetězec a literály true, null a false [12]. Většina programovacích jazyků umožňuje převod JSON dat na textový řetězec a zpět. V JavaScriptu je to metoda `JSON.stringify()`. Ať už jde tedy o ukládání uživatelských dat do cookies nebo do jiných úložišť popsaných v průběhu práce, nabízí se JSON jako ideální náhrada obyčejného textového řetězce, jelikož lze díky němu data uložit do jednotného a strukturalizovaného formátu. Výpis kódu číslo 1 ukazuje zápis dat v JSON formátu.

```
{
   "Cislo": 1,
   "Text": "Textovy retezec",
   "VnorenyObjekt": {
      "Boolean": true,
      "Null": null,
      "Pole": [1, 9, 9, 7],
   },
}
```

Výpis 1: Ukázka JSON objektu

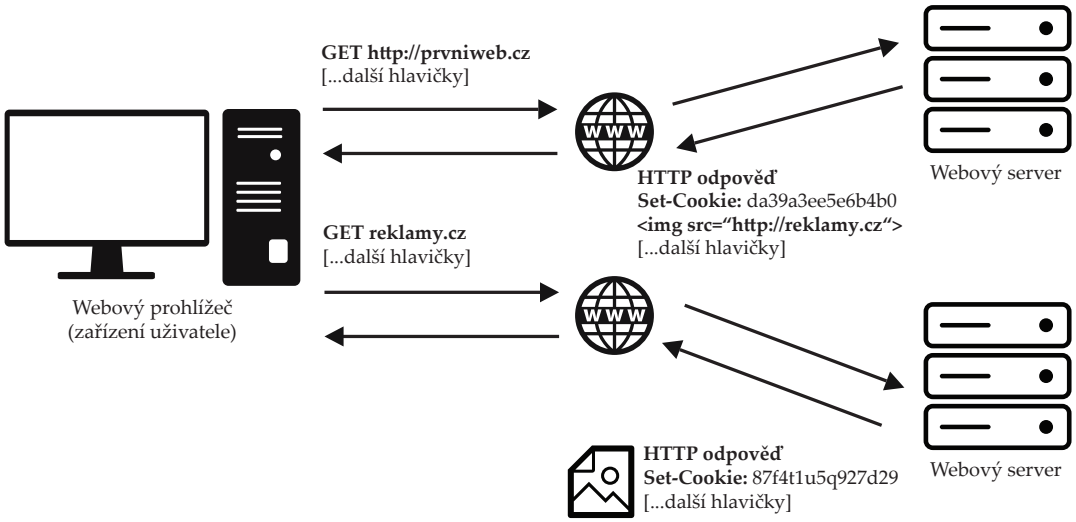
JavaScript je dnes používán drtivou většinou webových stránek a je tak nedílnou součástí WWW. Jeho poměrně aktuální verzi z roku 2015 podporují téměř všechny moderní prohlížeče (až na Internet Explorer) a verzi z roku 2009 pak údajně úplně všechny [13]. Uživatel může JavaScript v prohlížeči úplně vypnout nebo používat prohlížeč, který jej nepodporuje. To se děje například z toho důvodu, že chce webovým stránkám zamezit, aby mu zobrazovaly v JavaScriptu implementována vyskakovací okna, která často ruší při prohlížení webu. Nelze se tak s jistotou spolehnout na úspěšnost sledování uživatelů při jeho využití.

2.4 Same-origin policy

Nedávalo by smysl, aby ke cookies jedné webové stránky, respektive jedné doménové adresy, mohly přistupovat všechny ostatní existující stránky. Způsob, podle kterého se řídí přistupování ke cookies a jejich ukládání, určuje koncepce same-origin policy, což lze volně přeložit jako politika stejného původu. V principu říká, že obsah jednoho původu nemůže zasahovat do obsahu jiného původu. Totožný původ definuje jako takový, který má stejnou doménovou adresu serveru, protokol pro přístup k serveru a port, na kterém server běží (pokud je poskytnut). V praxi to znamená, že obsah jedné webové stránky má dovoleno číst a měnit pouze obsah sama sebe (stejného původu) a nemá dovoleno číst a měnit obsah stránky jiné [14].

2.5 Cookies prvních a třetích stran

Cookies jsou díky same-origin policy přístupné pouze z té webové stránky, ze které byly uživateli vytvořeny. Z tohoto důvodu se jeví, že uživatel nemůže být na jednom webu sledován jiným webem. Je tomu ovšem naopak. Cookies prvních stran jsou ty, které vytváří webová stránka přímo navštěvovaná uživatelem. V dnešní době ale webové stránky velmi často obsahují vsazené prvky z jiných webů, například: tlačítko „To se mi líbí“ od sociální sítě Facebook.com nebo video z Youtube.com. Tyto kusy kódů z jiných webových stránek zahrnuté v původní navštívené stránce se při návštěvě musí také načíst, a proto mohou v prohlížeči uživatele nepřímo vytvářet cookies, i když o tom dotyčný nemusí vůbec vědět, a především s tím souhlasit. Tento případ cookies se označuje jako cookies třetích stran. Webové stránky takto mohou sledovat aktivity uživatelů na všech ostatních webech, které mají reference na části jejich webu [15, 16]. Situace je podrobně znázorněna na obrázku 1.



Obrázek 1: Vznik cookies prvních a třetích stran (*Zdroj: vlastní zpracování*)

Uživatel navštíví webovou stránku prvniweb.cz obsahující prvky z reklamy.cz. Kromě cookies ze samotné stránky prvniweb.cz obdrží uživatel také cookies od reklamy.cz. Uživatel poté pokračuje na webovou stránku druhyweb.cz, která má v sobě také část z reklamy.cz. Prohlížeč odešle na server reklamy.cz totožnou cookie, kterou získal dříve, při návštěvě prvniweb.cz. Na straně serveru se nalezne shoda zaslané cookie, uživatel se identifikuje, a tím získá třetí strana, reklamy.cz, informace o aktivitách na obou navštívených webech.

Cookies prvních stran jsou prvoplánově užitečné především pro uživatele a jeho interakci s přímo navštěvovaným webovým serverem. Cookies třetích stran velmi často slouží ke sledování aktivit uživatele mezi webovými servery, a to především reklamními společnostmi. Nelze se pak divit, že cookies třetích stran vznesly obavy mezi uživatele internetu vztahující se k jejich soukromí a ochraně jejich dat. Této tématice se věnuje následující podkapitola. Dnešní webové prohlížeče však již standardně poskytují možnost vypnutí cookies třetích stran nebo celkové

vypnutí cookies, uživatel tak má do jisté míry kontrolu nad informacemi, které si o něm webové stránky pamatují.

2.6 Ochrana osobních údajů

World Wide Web Consortium (W3C), mezinárodní organizace tvořící standardy pro WWW, představila v roce 2002 pokus o sjednocení politiky soukromí napříč webovými prohlížeči a stránkami, a to v podobě W3C specifikace P3P 1.0 [17]. Účel P3P specifikace byl umožnit webovým stránkám sdělit jejich záměr o využívání cookies svým návštěvníkům, a to v určitém standardizovaném formátu. Stejně tak by na straně webového prohlížeče mohl uživatel přednastavit určité zásady, se kterými souhlasí. Zmíněný formát by prohlížeče uměly číst a automaticky by informovaly uživatele o tom, zda daná stránka splňuje uživatelovo nastavení o využívání cookies, či nikoliv. Uživatel by tak nemusel tyto informace pokaždé číst.

Webové prohlížeče od Microsoft (Internet Explorer a Microsoft Edge), jsou jediné populární prohlížeče, které kdy implementovaly P3P standardy. Prohlížeč Mozilla Firefox podporoval některé prvky, ale v roce 2007 z něho byl všechen tento zdrojový kód odstraněn. Od konce roku 2016, s počátkem operačního systému Windows 10, již Microsoft nepodporuje P3P v jeho výše zmíněných prohlížečích [18]. P3P specifikace byla v srpnu roku 2018 označena jako deprecated neboli zastaralou, především kvůli minimálnímu použití ze strany webových stránek a podle W3C by se neměla dále používat.

Dnes jsou cookies mezi uživateli internetu velmi známé, hlavně díky vyskakovacím oknům v podobě prohlášení webových stránek o jejich politice používání cookies. Evropská unie vydala v roce 2002 směrnici, která se zabývala zpracováním osobních údajů a ochrany soukromí v elektronickém odvětví, tudíž i cookies. Udávala webovým stránkám za povinnost informovat návštěvníky o zpracování jejich údajů prostřednictvím cookies, což v praxi znamenalo obdržet od uživatele souhlas s tímto zpracováním a to dříve, než se jakékoli cookies mohly uložit. Poslední nařízení říká, že uživatel může udělit souhlas webové stránce například zaškrtnutím políčka při její návštěvě, kde je zřetelně ukázáno, že návštěvník tímto projeví souhlas se zpracováním jeho údajů navrhovaným způsobem. Podle Úřadu na ochranu osobních údajů je dialog s upozorněním a informacemi o používání cookies na dané webové stránce příliš obtěžující. Pro seznámení uživatele s přítomností cookies má být postačující dodatečná informativní stránka, která je na webu dostatečně viditelná a dostupná. [19]

2.7 Nástupci cookies a důvody jejich vzniku

Prvotním cílem cookies bylo jednoduché ukládání relace uživatele s danou webovou stránkou, aby uživatel při opětovné návštěvě nemusel se stránkou interagovat v neměnných oblastech stále dokola. Velmi častým důvodem dnes však bývá přeprodávání informací o aktivitách uživatele na internetu, respektive informací o tom, které webové stránky navštěvuje a co vyhledává. Tyto informace mohou dále využít marketingové společnosti k zobrazování reklam cílovému uživateli,

které mu na základě jeho navštěvovaných webů nejlépe vyhovují. Ať už se jedná o monitorování uživatele z různých marketingových, statistických nebo jiných důvodů, je třeba mít spolehlivější řešení, než jakým jsou samotné cookies. Kvůli existenci cookies třetích stran se dnes uživatelé často snaží vyhnout tomu, aby byli sledováni. Mohou tak učinit zakazováním cookies přímo v prohlížeči, nebo používáním softwarů třetích stran (antiviry nebo pluginy). Webové stránky proto hedají nové metody sledování uživatelů, u kterých je ideálně vyloučené narušení ze strany uživatele, a tím mít korektní statistiky k různým účelům.

Tímto je uzavřena kapitola seznamující čtenáře se základními mechanikami a historií cookies a dalšími záležitostmi s nimi souvisejícími. V následující kapitole proběhne analýza různých způsobů identifikace uživatelů webu bez použití cookies.

3 Analýza

Tato kapitola se zabývá způsoby identifikace uživatelů webu, které jsou odlišné od cookies. Shrnující text T. Bujlowa a kolektivu z roku 2015, zabývající se metodami sledování uživatelů webu a možnou obranou proti nim, rozděluje metody do několika kategorií [20]. Tyto kategorie způsobů identifikování uživatelů webu budou převzaty a lehce upraveny pro strukturalizaci analýzy této práce a to následovně:

- **Metody založené na ukládání do paměti prohlížeče 3.1**
- **Metody založené na ukládání do úložišt pluginů 3.2**
- **Metody založené na webové mezipaměti 3.3**
- **Fingerprinting metody 3.4**
- **Ostatní metody 3.5**

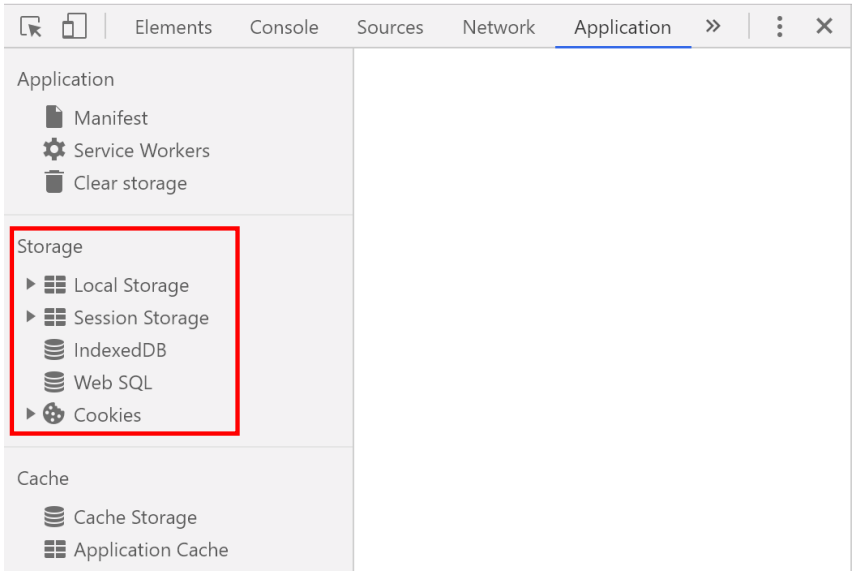
U jednotlivých metod bude kromě jejich definice a vysvětlení psáno také o možném druhu obrany proti nim a pokud to bude vhodné, tak se případně objeví i ukázka jejich použití ve formě kódu.

3.1 Metody založené na ukládání do paměti prohlížeče

V roce 2014 byla vydána W3C specifikace dlouho očekávané verze značkovacího jazyka HTML, HTML5 [22]. Tato specifikace přinesla obrovské množství novinek a vylepšení oproti předchozí verzi z konce devadesátých let minulého století. Adaptace HTML5 webovými stránkami byla poměrně rychlá a podpora v moderních prohlížečích je dnes velmi vysoká [23]. To je dáno spolehlivostí a univerzálností této verze a aktuálním trendem tvořit webové stránky primárně pro mobilní zařízení. Mobilní zařízení na rozdíl od počítačů totiž jen minimálně podporují pluginy do prohlížečů, určené například pro přehrávání videí, a proto je mnohem rozumnějším řešením sjednotit tyto platformy nativními HTML5 technologiemi.

V ohledu na sledování uživatelů webu přinesla tato verze především dvě nové možnosti ukládání dat do prohlížeče uživatele, které zastupují podobnou funkci jako cookies: Web Storage a Indexed Database (více informací níže). Většina moderních prohlížečů poskytuje tzv. *nástroje pro vývojáře*, což je GUI[3], ve kterém lze mimo jiné zobrazit obsah úložišť prohlížeče, přesněji cookies a HTML5 úložišť. Uživatel může spravovat zmíněná úložiště prohlížeče v tomto zobrazení, které je většinou přístupné stisknutím klávesy F12 nebo kombinací kláves Ctrl + Shift + I. Ukázka lze vidět na obrázku 2.

[3]Grafické uživatelské rozhraní je prostředek pro interagování s prvky zařízení pomocí grafického rozhraní.



Obrázek 2: Zobrazení úložišť v nástrojích pro vývojáře prohlížeče Google Chrome

3.1.1 Window.name

Objekt `window` reprezentuje okno v prohlížeči, přesněji řečeno každá záložka prohlížeče je reprezentována vlastním window objektem. Tento globální objekt je přístupný z JavaScript kódu a jedním z jeho potomků je objekt `document`, který byl zmíněn v kapitole 2.3. Objekt má mimo jiné také vlastnost `name`, primárně sloužící pro nastavení nebo získání jména záložky prohlížeče. Lze do ní uložit data ve formě textového řetězce a tyto data se při obnovení stránky nemažou [21].

3.1.2 HTML5 Web Storage

Asi nejpoužívanějším HTML5 úložištěm je Web Storage API a jeho Storage rozhraní, které umožňuje zvlášť manipulovat se sessionStorage a localStorage objekty webové stránky [24].

LocalStorage slouží pro perzistentní uložení dat v prohlížeči a přitom se zároveň se sessionStorage řídí same-origin policy (vysvětleno v kapitole 2.4). To se projevuje tak, že data uložená skrz určitou webovou stránku jsou opět přístupná pouze a jen ze stejné stránky. Přistupuje se k němu stejnojmenným objektem localStorage, který spadá pod window objekt. Uložené objekty jsou ukládány ve formě klíče a hodnoty. Tyto párové objekty se samy o sobě neodstraní, nemají totiž expirační dobu jako například cookies. Odstranit se dají explicitně buď webovou stránkou, nebo přímo uživatelem v GUI prohlížeče. Limit velikosti uložených dat je napříč dnešními prohlížeči nejméně 5 MB [25]. LocalStorage se díky zmíněným vlastnostem nabízí jako další možnost pro identifikaci uživatelů webu.

SessionStorage funguje obdobně jako localStorage s tím rozdílem, že data v něm nejsou uložena perzistentně, ale jen po dobu jedné relace, tzv. *session*, pro danou záložku v prohlížeči. Data jsou ze sessionStorage odstraněna po zavření záložky a při jejím obnovení se stejně jako u

localStorage nesmažou. Rozdíl v syntaxi obou těchto Web Storage objektů je pouze v klíčovém slovu sessionStorage namísto localStorage. Výpis číslo 2 ukazuje uložení, přečtení a smazání dat z localStorage.

```
// Uložení JSON objektu do localStorage
localStorage.setItem("WebUser", JSON.stringify({ id: 1, flash: true }));
// Získání hodnoty z localStorage
localStorage.getItem("WebUser");
// Odstranění hodnoty z localStorage
localStorage.removeItem("WebUser");
```

Výpis 2: Ukázka použití localStorage

3.1.3 HTML5 Indexed Database

Celým názvem Indexed Database API 2.0 je objektově orientovaná databáze uchovávající informace v podobě klíčů a objektů, řídící se Same Origin Policy stejně jako Web Storage [26]. Databáze je indexovaná, a také díky tomu je na rozdíl od Web Storage vhodné ji použít pro větší množství dat. Objekt uložený pod klíčem nemusí být striktně pouze textový řetězec, jako tomu je u předchozích zmíněných metod ukládání do paměti, může se jednat o téměř jakýkoli tradiční datový typ [27]. Předchůdce této databáze byla dřívější Web SQL Database API, kterou označilo W3C za zastaralou a za její přijatou náhradu je aktuálně považována právě Indexed Database API 2.0.

Indexed Database funguje tak, že každý databázový úkon je požadavek, kterému náleží handlery událostí, skrz ně můžeme v kódu databázi používat. Druhou možností, jak databázi používat, je využití IndexedDB Promised knihovny, která překládá původní požadavky na *promise* konstrukci. Princip fungování zůstává stejný, pouze syntaxe se lehce liší [28].

```
window.indexedDB = window.indexedDB || window.mozIndexedDB || window.
    webkitIndexedDB || window.msIndexedDB;
    // Prohlížeč nepodporuje IndexedDB
    if (!window.indexedDB) { };
    var request = window.indexedDB.open('NewDB', 1);
    var db, transaction, store;
    // Kód spouštěný při vytvoření nové databáze
    request.onupgradeneeded = function () {
        db = request.result;
       // Vytvoření store objektu
        store = db.createObjectStore('NameStore', { keyPath: 'id' });
        // Uložení hodnoty do databáze
        store.put({ name: 'Leoš Mareš' });
    };
    // Pokud je navázání spojení s danou databází úspěšné
    request.onsuccess = function () {
            db = request.result;
            transaction = db.transaction('NameStore', 'readwrite');
            store = transaction.objectStore('NameStore');
            // Pokus o přečtení záznamu podle id
            var name = store.get(0);
            // Kód následující při úspěšné přečtení
            name.onsuccess = async function () { };
            // Uzavření databáze při úspěšné transakci
            transaction.oncomplete = () => {
                db.close();
            };
    };
    // Pokud dojde k chybě při navázání spojení s databází
    request.onerror = function () { };
```

Výpis 3: Ukázka použití IndexedDB

3.2 Metody založené na ukládání do úložišť pluginů

NPAPI je dnes již zastaralé aplikační rozhraní s velmi malou podporou, které slouží ke tvorbě pluginů pro webové prohlížeče. S příchodem páté verze HTML byly pluginy radikálně vytlačovány a nahrazovány použitím nových, nativních HTML5 technologií. Kvůli jejich stále klesající, mizivé podpoře lze dnes jejich zneužití pro sledování uživatelů webu označit za neúčinné, ale pro účely analýzy budou uvedeny.

3.2.1 Flash Cookies

Adobe Flash Player je plugin do webových prohlížečů od společnosti Adobe Systems, který primárně umožňuje zobrazovat animace, hry a videa vytvářené nad platformou Adobe Flash. V situaci, kdy si Adobe Flash potřebuje uložit potřebná data, ukládá je do počítače uživatele v podobě LSO souborů, také známých jako Flash cookies. Ty obsahují data například pro načítání obsahu videa, ale mohou být využity ke stejnému účelu jako HTTP cookies. Jejich kapacita je 100 KB a nemají implicitně danou dobu expirace, proto je lze smazat pouze explicitně.

Zneužití Flash cookies pro sledování uživatelů webu vyžaduje vytvoření identifikátoru uživatele na straně serveru a jeho následné uložení do LSO souboru. Velkou výhodou oproti HTTP cookies je fakt, že Flash cookies jsou přístupné ze všech prohlížečů nainstalovaných v zařízení uživatele, jelikož každá nainstalovaná Flash aplikace, například Flash Player, sdílí adresář úložiště dat se všemi ostatními. Každý prohlížeč má proto k dispozici stejná data a díky tomu lze uživatele sledovat napříč prohlížeči.

Odstranění Flash Cookies běžnými uživateli webu je velmi nepravděpodobné, protože mezi veřejností existuje jen malé povědomí o této problematice. Navíc v GUI prohlížečů není nabídka, která by umožňovala jejich smazání a při obyčejném vymazání cookies se nesmažou, jelikož se do prohlížeče neukládají. Odstranit se dají několika způsoby. Manuálním smazáním obsahu adresáře jejich úložiště, dále v nastavení zabezpečení počítače pro Flash Player a nebo v nastavení pro Flash Player na oficiální stránce společnosti Macromedia vlastnící Adobe Systems. Existují také pluginy, které toto umožňují přímo z prohlížeče. Při používání prohlížeče v režimu anonymního prohlížení dojde k zamezení ukládání Flash Cookies [29].

Průzkum z roku 2011, zabývající se použitím LSO souborů pro znovu vytvoření smazaných cookies, zjistil, že 9 ze 100 v té době nejpopulárnějších webových stránek ukládalo do těchto souborů unikátní obsah [30]. Z průzkumu tedy vyplývá, že maximálně 9 % z těchto 100 webů mohlo používat Flash Cookies pro sledování uživatelů, a to pro znovuvytvoření smazaných cookies nebo přímo pro samotné rozpoznání uživatele na základě unikátního identifikátoru. Tato hodnota sice není úplně zanedbatelná, ale počet webových stránek používajících Adobe Flash od roku 2011 postupně klesal a podle webu *w3techs.com* jej dnes využívá jen 3,6 % webů [31]. Adobe Systems v roce 2017 oznámila ukončení podpory jejich platformy na konci roku 2020 [32], a to hlavně kvůli její stále klesající popularitě. Ve stejnou dobu bude ve verzích Google Chrome a ostatních moderních prohlížečů úplně odstraněn Adobe Flash [33].

3.2.2 Microsoft Silverlight

Silverlight je platforma od společnosti Microsoft se stejnojmenným pluginem do prohlížečů, který byl vydán v roce 2007 pravděpodobně ve snaze konkurovat tehdy světově populární platformě Adobe Flash. Podobně jako Adobe Flash poskytuje úložiště dat na zařízení uživatele v podobě tzv. *Isolated Storage*, a to o výchozí kapacitě 1 MB pro jednotlivou webovou stránku, přičemž lze kapacitu navýšit [34]. Zneužití tohoto úložiště pro sledování uživatele spočívá ve stejném postupu jako je tomu například u Flash Cookies, a to vytvořením identifikátoru uživatele a jeho následným uložením do Isolated Storage pro pozdější identifikaci stejného uživatele. Stejně jako v případě Flash Cookies se data do Isolated Storage neukládají při používání režimu anonymního prohlížení. V dnešní době je Microsoft Silverlight ještě zanedbatelnější hrozbou pro anonymitu návštěvníků webu, než jakou je Flash Player, a to z důvodů vysvětlených níže.

Microsoft v roce 2013 oznámil ukončení vývoje Microsoft Silverlight vyjma opravování chyb a celkové ukončení podpory stanovil na říjen 2021 [35]. Prohlížeče Google Chrome, Mozilla Firefox a Opera v roce 2015 zrušily podporu Microsoft Silverlight a Microsoft Edge jej paradoxně dokonce nikdy nepodporoval. Jeho využití webovými stránkami je menší než 0,1 % a postupně se blíží nule [31].

3.2.3 Ostatní

Existuje mnoho dalších pluginů, které také disponují podobným typem úložiště jako Flash nebo Silverlight, mezi další nejznámější patří Java. Jejich míra používánosti je dnes ale ještě menší než u předchozích dvou a jsou tak již irelevantní [31].

3.3 Metody založené na webové mezipaměti

Součástí webových prohlížečů je mezipaměť prohlížeče, tzv. web cache, sloužící jako dočasné úložiště částí webových stránek (dat). Při návštěvě webové stránky si prohlížeč do web cache ukládá kopii dat, které bylo nutné stáhnout pro úplné načtení webu. Když se uživatel vrátí na danou stránku, prohlížeč si díky web cache pamatuje, které části stránky již byly staženy při dřívější návštěvě. Pod podmínkou, že se nezměnila požadovaná část webu, ji nemusí prohlížeč opět stahovat, protože pro její načtení stačí použít kopii uloženou ve web cache.

Na základě této funkcionality je stránka schopna detekovat, jestli uživatel navštívil stránku poprvé, což se projeví nutností stáhnout ze stránky data a uložit je do cache prohlížeče. Pokud se části stránky načtou z mezipaměti (není je tedy potřeba stahovat), uživatel uskutečnil opakovanou návštěvu webu.

3.3.1 HTTP hlavičky ETag a If-None-Match

HTTP GET (a také HEAD) požadavky mohou být z obyčejných požadavků změněny na podmínkové požadavky, pokud obsahují alespoň jednu podmínkovou hlavičku. Výsledná odpověď na

požadavek pak závisí na tom, jestli dojde ke splnění podmínky hlavičky, či nikoliv. Ke sledování uživatelů webu se nabízejí dvě takovéto hlavičky, a to konkrétně If-None-Match a If-Modified-Since a k nim náležící hlavičky HTTP odpovědí ETag a Last-Modified (v tomto pořadí).

Jak již bylo vysvětleno na začátku této kapitoly 3.3, pokud nedošlo ke změně nějaké části webu, server nemusí posílat odpověď se všemi daty webu. K tomuto rozeznání změny lze využít zmiňovanou dvojici HTTP hlaviček ETag a If-None-Match. **ETag** nese identifikátor *entity-tag* určité verze obsahu a je zasílán v odpovědi serveru. Tento entity-tag je lépe řečeno identifikátor interpretace dat a slouží jako tzv. validátor pro rozlišení různých reprezentací stejného obsahu, přičemž jeho formát není jasně stanoven a jeho maximální délka také ne [36]. Proto se entity-tag při jeho využití ke sledování uživetelů nejčastěji generuje pomocí hašovacích funkcí[4], například *MD5* nebo *SHA*.

Pro srovnání takovýchto entity-tagů se využívá **If-None-Match** hlavička, která jak již bylo zmíněno, dělá z obyčejného GET požadavku podmínkový požadavek. Tato hlavička kontroluje shodu entity-tagu zaslaného prohlížečem s aktuálním entity-tagem daného obsahu na serveru. Pokud se hodnoty neshodují, je podmínka splněna a server v odpovědi zašle stavový kód HTTP 200 OK s daným obsahem webu v těle odpovědi. V opačném případě se jedná o stejnou interpretaci dat a server odpoví stavovým kódem 304 Not Modified s prázdným tělem odpovědi [37]. Výše popsané fungování znázorňují obrázky 6 a 7.

Stavový kód 304 Not Modified značí, že prohlížeči není potřeba znovu zasílat požadovaný obsah webové stránky, jelikož jeho uložená verze v cache prohlížeče se od aktuální verze na serveru neliší [38]. Namísto toho se obsah načte z mezipaměti prohlížeče a jde tak o druh přesměrování. [39]

3.3.2 HTTP hlavičky Last-Modified a If-Modified-Since

Obdobným způsobem funguje kombinace hlaviček **If-Modified-Since** a **Last-Modified**. Server odpovídá hlavičkou Last-Modified obsahující datum a čas okamžiku, kdy byla část webu na serveru naposled upravena [40]. Prohlížeč zasílá v If-Modified-Since čas a datum poslední úpravy své verze uložené v cache. Porovnává se, zda je jedna z hodnot hlaviček starší, nebo novější, než ta druhá [41]. Hlavičky nemusí striktně nést pouze formát data a času, a proto je lze využít pro uchovávání identifikátoru uživatele stejně jako u ETag a If-None-Match. [39]

3.3.3 Shrnutí metod založených na mezipaměti

Použití zmíněných metod za účelem sledování lze při návštěvě webu jen velmi těžko zpozorovat, jelikož za sebou například na rozdíl od cookies nenechávají tolik viditelné stopy. Uživatel se proti nim může bránit pravidelným mazáním obsahu web cache při každé návštěvě webu, u kterého chce mít jistotu, že nebude sledován. Radikálnějším řešením je úplné vypnutí web cache, čímž

[4] Jedná se o algoritmy, které na základě vstupních dat vytvoří kompaktnější interpretaci, tzv. *hash*, většinou jako výstupní textový řetězec.

ale dojde k poměrně výraznému poklesu v rychlosti odezvy prohlížeče, jelikož každá návštěva webu se bude vždy považovat za první a potřebná data se budou muset vždy opětovně stahovat.

3.4 Fingerprinting metody

Webový prohlížeč poskytuje poměrně velké množství informací o svém nastavení a o zařízení, ze kterého návštěva webu probíhá. Jedná se například o název grafické karty, nainstalované pluginy, preferované jazyky uživatele, operační systém zařízení, seznam nainstalovaných fontů, rozlišení obrazovky a mnoho dalších. Tyto informace jsou individuálně nic neříkající, ale dohromady tvoří otisk prohlížeče a zařízení, takzvaný *fingerprint*, který může být do jisté míry unikátní a může proto sloužit ke sledování a identifikaci dotyčného. Na rozdíl od kterékoliv jiné metody navíc může fungovat v režimu anonymního prohlížení. Identifikace uživatelů webu s pomocí těchto otisků se nazývá *fingerprinting* a jedná se o experimentální, ale stále populárnější techniku.

Podstatná část informací, tvořících dostatečně unikátní otisk prohlížeče, se získává pomocí spuštění JavaScript kódu na straně klienta, tedy v jeho prohlížeči. Úspěch fingerprinting metod využívajících JavaScript proto závisí na tom, že není v prohlížeči vypnutý (například záměrně, samotným uživatelem). Zbylá část dat většinou pochází z hlaviček obyčejného HTTP požadavku, jež jsou serveru poskytnuty při výměně informací s prohlížečem. Jak bude v kapitole 5 řečeno, tento typ informací nemá tak velkou identifikační hodnotu jako hodnoty získané JavaScriptem. Na rozdíl od JavaScript fingerprinting metod, jejichž přítomnost již dnes z části lze odhalit (viz. kapitola 6), je však zneužívání informací získávaných z HTTP požadavků prakticky nemožné vystopovat. Další existující potenciální informace mohou existovat a třeba se o jejich vypovídající hodnotě pouze neví. Nejzajímavější fingerprinting metody, některé zároveň použity v implementaci aplikace, jsou detailně popsány v kapitole 4.1.

Podobně jako tomu bylo u metod sledování založených na webové cache, lze fingerprinting také obtížně odhalit a obrana proti němu je mnohem komplikovanější. Snaha zvýšit ochranu svého soukromí před fingerprintingem může být totiž kontraproduktivní. Provedením dodatečných opatření týkajících se prohlížeče jako například vypnutí cookies, instalace pluginů, atd. se však naopak může zvýšit šance být úspěšně identifikován. Prohlížeč uživatele se totiž s dodatečnými změnami více odliší od většiny prohlížečů a uživatel se tak potenciálně sám stává jednodušším terčem, jelikož je jeho otisk o to unikátnější.

Dosavadní výzkum

G. Acar a kolektiv ve své práci z roku 2014 otestovali výskyt fingerprintingu na 100 000 nejpopulárnějších webových stránkách internetu a zjistili výskyt větší než 5 % [42]. Další práci s obdobným cílem publikoval S. Englehardt a A. Narayanan v roce 2016, kde otestovali výskyt fingerprintingu mezi jedním milionem nejpopulárnějších webů internetu [43]. Jejich výsledky například odhalují použití tzv. *canvas fingerprintingu* (více v kapitole 4.1.1) u 1,6 % testovaných webů.

Jedny z dosud nejrozsáhlejších a nejznámějších publikací zaobírajících se problematikou fingerprintingu jsou projekty Panopticlick z roku 2010 [44] a AmIUnique z roku 2016 [45]. P. Eckersley v projektu Panopticlick vůbec jako první ukázal potenciál fingerprintingu a odstartoval tím zájem o jeho další výzkum a využití. Výsledky postavené na celkem 470 161 posbíraných otiscích ukazují, že náhodně vybraný fingerprint se bude opakovat pouze v jednom případu z 286 777. Časem se otisk prohlížeče mění, a proto Eckersley demonstroval jednoduchý algoritmus schopný znovu identifikovat fingerprint i přes to, že nebyl totožný s žadným z předchozích. Tento algoritmus srovnával podobnost textových řetězců (fingerprintů) a v případě alespoň 85 % shody identifikoval řetězce jako jeden. Jeho úspěšnost byla znepokojivých 99,1 %. Nutno podotknout, že v případě nepřítomnosti Flash nebo Java pluginu se algoritmus o uhádnutí fingerprintu vůbec nepokoušel (nejspíš z důvodů mnohme menší přesnosti) a množství takovýchto případů bylo nezanedbatelných 35 %.

P. Laperdrix a kolektiv publikovali výsledky získané z projektu AmIUnique v podobné formě jako tomu bylo u Panopticlick. Počet získaných fingerprintů byl 118 934 a oproti svému předchůdci obohatili sběr dat o další moderní techniky. Nejspíš nejzásadnější přínos bylo zhodnocení účinnosti tzv. *canvas fingerprintingu* v praxi. Tuto metodu sice dříve popsali K. Mowery a H. Shacham v roce 2012 [46] jako nový potenciální způsob fingerprintingu, ale její skutečnou schopnost získat unikátní hodnoty otestovali pouze u 300 fingerprintů. Výsledky z AmIUnique výrazně přispěly také díky faktu, že jejich analýzu tvořily z 11 % mobilních zařízení a prokázala se u nich zásadní unikátnost 81 %.

Oba zmíněné projekty svá data sbírají na vlastních webových stránkách, kde své návštěvníky seznamují s účelem webu ještě před tím, než proběhne samotný sběr dat. Teprve po získání jejich souhlasu proběhne sběr dat a následné zobrazení informací týkajících se otisku jejich prohlížeče. To znamená, že u obou projektů jsou návštěvníci se situací dopředu obeznámeni a mohou se proto pokoušet měnit nastavení svých prohlížečů, ať už ve prospěch unikátnějšího fingerprintu, nebo naopak. Můžeme tedy s jistotou říct, že výsledky obou publikací vycházely z částečně zaujatých dat, což ostatně oba zdroje zmiňují, a je potřeba je brát s rezervou.

3.5 Ostatní metody

V této podkapitole budou jmenovány některé další existující způsoby identifikování uživatelů, které nebylo možné jednoznačně zařadit do některé z předchozích kategorií, nebo jsou výsledky jejich použití dosud příliš nekonzistentní.

3.5.1 Evercookies

V roce 2010 ukázal Samy Kamkar své JavaScript API nazvané Evercookie, mající za účel demonstrovat velmi perzistentní cookies, evercookies [67]. Cíl evercookie je zařídit, aby uživatel nebyl schopen jednoduše odstranit cookies ze svého přohlížeče. Toto zajišťují evercookies, také známé jako supercookies nebo zombie cookies. Jedná se o kombinaci až osmnácti odlišných me-

tod ukládání cookies v prohlížeči uživatele pro jeho následnou identifikaci na webu např.: HTTP cookies, HTTP Etag, HTML5 metody a window.name. Na rozdíl od obyčejných HTTP cookies uložených právě a jen v cookie storage prohlížeče, jsou evercookies založené na ukládání stejných cookie dat na různá místa v prohlížeči. Pokud uživatel vymaže jedno nebo více úložišť cookies, evercookie má ta samá cookie data stále uchována na mnoha dalších místech. Evercookie navíc umí zjistit, jestli došlo k odstranění nějakých druhů cookies, a následně je znovu vytvořit v dalším požadavku. Díky tomu není jednoduché se bez softwarů třetích stran takových cookies zbavit, uživatel o jejich přítomnosti většinou ani neví. Evercookies mohou být také přenositelné mezi prohlížeči, například při používaní již zmíněného Adobe Flash Playeru a jeho flash cookies ve více prohlížečích.

3.5.2 Úložiště userData od prohlížeče Internet Explorer

Prohlížeč Internet Explorer představil vlastní úložiště zvané *userData*, které nesklidilo úspěch a nyní je již označeno jako zastaralé. Fungovalo technikou přidání tzv. *behavior* CSS vlastnosti některému z DOM objektů, čímž bylo tomuto objektu umožněno přístupovat ke zmíněnému úložišti [68]. Navzdory faktu, že Microsoft označil také samotnou *behavior* CSS[5] vlastnost jako zastaralou, v některých verzích prohlížeče toto úložiště stále funguje [69].

[5]Kaskádové styly (anglicky Cascading Style Sheets) slouží k přizpůsobení zobrazení obsahu webové HTML stránky.

4 Návrh a implementace

Cílem aplikace je využít a otestovat alternativní metody pro identifikování uživatelů webu. Na základě analýzy z předchozí kapitoly budou implementovány pouze dvě nejzajímavější a nejperspektivnější řešení: fingerprinting a mezipaměť prohlížečů. V další kapitole bude otestována schopnost těchto dvou metod v praxi (mezi uživateli) napříč moderními prohlížeči.

Aplikace je webová, tedy typu klient server. Data o uživatelích jsou ukládána do databáze, kterou tvoří jedna tabulka obsahující záznamy o uživatelích ve formě logu.

4.1 Klientská část

Klientská část má za úkol posbírat data o uživateli navštěvující web a pomocí AJAX [6] poslat výsledky na server, kde se dále zpracují a uloží do databáze.

V klientské části aplikace byla použita externí open-source knihovna `ua-parser`, sloužící k parsování User-Agent textového řetězce [47]. Knihovna již sice není udržovaná, ale pro účely této práce je dostačující. Zbytek kódu klientské části je pouze čistý JavaScript, především z důvodu zajištění co nejvyšší zpětné kompatibility mezi různými prohlížeči.

4.1.1 Fingerprinting

Implementace použitých technik se opírá o již zavedené postupy. Inspirací byly především publikace projektů Panoptioclick a AmIUnique, zmíněných v kapitole 3.4, a také z open-source knihovny fingerprintingjs2 [48]. Fingerprinting data posbíraná z JavaScriptu se pomocí HTTP protokolu zasílají na server a společně s fingerprinting daty z HTTP hlaviček se uloží do databáze.

Nepoužité metody

Výsledky práce Panopticlick i AmIUnique jednoznačně prokázaly vysokou schopnost pluginu Adobe Flash Player výrazně zvýšit unikátnost fingerprintů a spolehlivěji tak identifikovat uživatele. V kapitole 3.2.1 byl však zhodnocen jeho podíl na internetu v dnešní době a z těchto důvodů nebude v implementaci fingerprintingu zahrnut. To stejné platí pro jakékoli jiné pluginy, jako například Microsoft Silverlight nebo Java.

User-Agent

User-Agent hlavička je nezbytná součást HTTP požadavku. Ve většině případů přináší největší zdroj informací ze všech jmenovaných fingerprinting metod v této kapitole. Obsahuje informace

[6] Jde o použití *XMLHttpRequest* JavaScript objektu k asynchronnímu zásílání a získávání informací mezi klientem a serverem.

o zařízení a prohlížeči klienta, nejčastěji jimi jsou: název a verze prohlížeče, název a verze operačního systému, název a verze softwaru prohlížeče. Příklad User-Agent řetězce z prohlížeče Google Chrome a systému Windows 10: `Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36`. V případě mobilních zařízení obsahuje řetězec dokonce přesný marketingový název modelu nebo alespoň jeho technické označení.

Obsah User-Agent hlavičky se v aplikaci získává s pomocí JavaScriptu, přečtením vlastnosti `navigator.userAgent`. Přístupný je ale také ze strany serveru v požadavku klienta. Jednotlivé informace z řetězce se v aplikaci parsují pomocí knihovny *ua-parser*, zmíněné na začátku kapitoly. Knihovna funguje na principu srovnávání řetězce s předem definovanými regulárními výrazy[7]. Konzistentně správné výsledky parsováním User-Agent řetězce jsou samozřejmě podmíněny aktualizováním takové knihovny, protože mobilních zařízení stále přibývá a formáty User-Agent hlavičky se mohou měnit. Nicméně pomocí této knihovny lze potenciálně získat následující informace:

- **název a verze operačního systému**
- **název a verze prohlížeče**
- **název a verze softwaru prohlížeče**
- **typ zařízení (stolní počítač, mobilní zařízení, tablet a několik dalších)**
- **oficiální název modelu zařízení (dostupný například u mobilních zařízení)**
- **oficiální název prodejce zařízení**

Pokud se nějakou z informací nepodaří zjistit, má hodnotu prázdného textového řetězce.

Rozlišení zařízení

Dnešní trh nabízí rozsáhlé spektrum monitorů, notebooků a mobilních zařízení, u nichž se mohou vlastnosti displeje značně lišit, a pro účely fingerprintingu jsou tyto informace velmi identifikující. Ve výsledné aplikaci je využito rozlišení displeje zařízení a poměr jeho pixelů.

Rozlišení displeje zařízení, respektive jeho fyzickou výšku a šířku, lze získat z vlastností `height` a `width` objektu `Screen` (spadajícího pod objekt `Window`) [51]. Pokud se však změní monitor, ve kterém je prohlížeč spuštěn, tak se tyto hodnoty také změní. Toto může nastat u notebooků, ale také u stolních počítačů, a je proto třeba to brát v potaz. U mobilních zařízení je na druhou stranu velmi malá pravděpodobnost připojení externího monitoru, a proto se nepředpokládá změna rozměrů jejich displeje (kromě změny rotace o 90 a 180 stupňů).

[7] Známé jako *regex*, je posloupnost znaků definující vyhledávaný řetězec

Poměr pixelů displeje definuje poměr mezi rozlišením fyzických pixelů a rozlišením CSS pixelů [52] displeje daného zařízení. Toto číslo lze získat z vlastnosti `devicePixelRatio` objektu `Window`. U obyčejných monitorů bývá toto číslo 1 a u mobilních zařízení je z pravidla desetinné, například 2.625. Tento poměr pixelů je potřebný u mobilních zařízení, pro které výše popsaný způsob získání rozlišení sám o sobě nefunguje správně. Například displej od Xiaomi Mi A3 má specifikované rozlišení 1560x720 pixelů. Poměr pixelů má hodnotu 2, `window.screen.width` má hodnotu 780 a `window.screen.height` má hodnotu 360. Po vynásobení výšky a šířky poměrem pixelů získáme specifikované rozlišení. U zařízení, kde je poměr pixelů desetinné číslo, je vypočítaná hodnota rozlišení méně přesná. Pro účely fingerprintingu je toto však irelevantní, jelikož je pouze důležité získat konzistentní výsledky pro stejné zařízení.

Seznam pluginů

Seznam pluginů v prohlížeči je další informace využitelná pro fingerprinting, čemuž se ale moderní prohlížeče již začínají bránit. Seznam pluginů mohou využít například vývojáři, když potřebují zjistit existenci pluginu v prohlížeči uživatele. Na základě této informace pak uživatelům s chybějícím pluginem například zobrazí alternativní obsah nebo jim doporučí si plugin nainstalovat. Rozhraní `Navigator` poskytuje seznam pluginů a jejich vlastností v prohlížeči, a to zavoláním `navigator.plugins` [57].

Seznam fontů

Seznam nainstalovaných fontů v systému patří mezi vlastnosti prohlížeče s nejvíce různorodými výsledky. Prohlížeče sice tento seznam nativně neposkytují, jsou však jiné cesty, jak jej získat. Některé pluginy, například Adobe Flash Player, umí přečíst nainstalované fonty v systému a dokonce i v pořadí specifikovaném daným systémem. Výčet stejných prvků v jiném pořadí znamená vyšší neurčitost, a tedy unikátnější fingerprint. Jak bylo však řečeno na začátku této kapitoly, řešení s pomocí pluginů nebudou implementována. Neseřazený pseudo seznam nainstalovaných fontů lze získat také využitím CSS a JavaScriptu, tzv. *font probing*, a právě toto řešení je ve výsledné aplikaci použito. Část řešení poprvé ukázali N. Nikiforakis a kolektiv ve své publikaci z roku 2013 [58], dále bylo řešení vylepšeno v projektu AmIUnique a open-source knihovně fignerprintingjs2, z nichž se při implementaci čerpalo.

Metoda je schopna rozhodnout o existenci dotazovaného fontu v systému. Je tedy potřeba mít předem definovaný seznam fontů, ve kterém metoda dokáže nalézt ty, které se v systému vyskytují. Vyskytující se fonty pak tvoří seznam vhodný pro fingerprinting. Existence daného fontu v systému se otestuje několika kroky. Na začátku se vytvoří HTML element obsahující jednotný textový řetězec o jednotné velikosti. Obsah řetězce je vhodně zvolený tak, aby jeho znaky dokázaly odhalit všechny rozdílnosti mezi fonty, a zároveň aby počet znaků byl co nejmenší (pro co možná nejmenší, výkonostní požadavky). Velikost písma musí být dostatečná pro zachycení velmi malých změn v pixelech při vykreslení textu. Tento text se nejprve vykreslí s fontem,

který bývá mezi prohlížeči používán jako tzv. fall-back[8] a jeho rozměry (výška a šířka) se uloží. Rozměry textu získáme jednoduše z vlastností HTML elementu `offsetWidth` a `offsetHeight`, jelikož platí, že rozměry HTML elementu se přizpůsobí velikosti textu v něm. Potom textu nastavíme takový font, jehož existenci chceme zjistit, a rozměry textu se opět změří. V případě, že požadovaný font není v systému přítomen (byl zvolen fall-back font), vykreslený text bude mít v obou případech stejné rozměry, protože má nastavenou vždy stejnou velikost. Pokud je ovšem velikost textu s použitím testovaného fontu jiná než s použitím fall-back fontu, zamená to, že se font podařilo vykreslit a nachází se v systému. Tímto způsobem otestujeme celý požadovaný seznam.

Metoda má svá úskalí. V reálném scénáři nemůže počet otestovaých fontů přesáhnout určitou hranici, kdy doba exekuce kódu příliš zvýší odezvu webu. Předdefinované fonty tedy mohou obsahovat pouze určitý výběr, čímž se částečně ztrácí potenciál z hlediska fingerprintingu.

Canvas fingerprinting

Kombinací několika dílčích metod patří canvas fingerprinting mezi jedny z nejvíce identifikujících a nejstabilnějších způsobů fingerprintingu. Principem je vykreslování obrázku v prohlížeči uživatele s použitím totožných, neměnných instrukcí. Díky kombinaci rozdílů v operačním systému, prohlížeči a hardwaru zařízení (především ovladač grafické karty) se vykreslený obrázek může lišit. Rozdíly ve výsledném obrázku pak slouží jako informace identifikující uživatele.

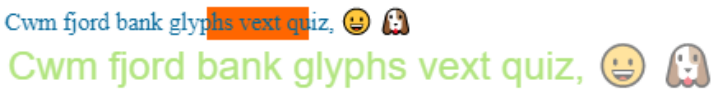
Implementace canvas fingerprintingu v aplikaci je převzata z řešení, které poprvé ukázal Acar a kolektiv a později vylepšil projekt AmIUnique. Obrázky 3, 4 a 5 ukazují výslednou grafiku vykreslenou na třech různých operačních systémech. Samostatně byla použita také upravená verze této implementace z open-source knihovny fingerprintingjs2, ve které jsou dále obohaceny parametry vykreslování (více v kapitole 5).

HTML element `canvas` poskytuje vykreslovací 2D rozhraní (dále jen canvas context) umožňující vytvářet a měnit grafiku, například texty a tvary [59]. Pomocí canvas contextu se vykreslí dvakrát stejný text, ale pokaždé s jiným fontem, barvou a velikostí. Vhodným textem je dokonalý pangram[9], protože umožní zachytit rozdíly ve všech písmenech abecedy, a zároveň má co nejmenší délku. V aplikaci je použit dokonalý pangram anglického jazyka, který se například používá na internetu pro reprezentaci typografie. Stejný font se na různých systémech může nepatrně lišit, což způsobí rozdíl ve výsledném canvas fingerprntu. Pro první text se použije neexistující fall-back font pro zaručení použití výchozího fontu operačního systému. Toto zaručíme tak, že od prohlížeče požadujeme font s vymyšleným, neexistujícím názvem. Pro druhý text se použije font Arial.

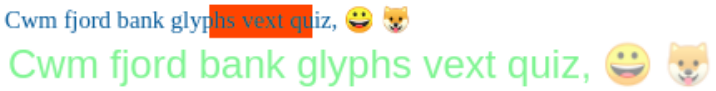
[8] Náhradní možnost pro případy, kdy preferovaná možnost není k dispozici.

[9] Věta obsahující všechny znaky dané abecedy. Dokonalý pangram obsahuje každé písmeno abecedy právě jednou.

Posledními dvěma znaky textu jsou tzv. emojis, což jsou piktogramy používané například pro vyjádření emocí. Emojis byly přidány do Unicode standardu a každé z nich je tedy reprezentováno unicode znaky. Vývojáři dnes proto zahrnují řešení jednotlivých emojis v implementaci fontů. Díky tomu se jejich grafická podoba může na různých zařízeních lišit, a jejich použití je proto vhodné pro fingerprinting, zejména na rozlišení rozdílů mezi operačními systémy a výrobci zařízení. Použití emojis má proto největší potenciál u mobilních zařízení, kde existuje velká rozmanitost výrobců.



Obrázek 3: Vykreslený canvas fingerprinting obrázek na Windows 10



Obrázek 4: Vykreslený canvas fingerprinting obrázek na Ubuntu 18.04 LTS

Obrázek 5: Vykreslený canvas fingerprinting obrázek na Android 10

Pro jednoduché porovnávání rozdílů mezi obsahem canvas elementu (na úrovni pixelů) existují příhodné metody. Canvas context disponuje metodou `getImageData()`, která pro daný canvas vrací objekt `ImageData`. Ten se skládá z RGBA celočíselných hodnot každého vykresleného pixelu. Ze samotného canvas objektu lze použitím metody `toDataURL()` získat reprezentaci celého obsahu canvas elementu v Base 64 kódování. Z těchto dat se nakonec vytvoří hash, který reprezentuje výslednou hodnotu canvas fingerprintingu daného uživatele.

WebGL informace

WebGL API je další rozhraní canvas elementu vhodné pro vykreslování grafiky [55]. Mimo jiné odhaluje také dvě konstanty s informacemi od ovladače grafické karty zařízení, a to skrz *WEBGL_debug_renderer_info* rozšíření [56]. Konstanta `Vendor` sděluje název společnosti zodpovědné za implementaci WebGL daného prohlížeče. Například u Google Chrome je touto hodnotou `Google Inc`. Druhá konstanta, `Renderer`, nese název modelu grafické karty, a také informaci identifikující operační systém zařízení. Příklad `Renderer` hodnoty z operačního systému Windows 10: `ANGLE (NVIDIA GeForce GTX 1080 Ti Direct3D11 vs_5_0 ps_5_0)`.

Tyto konstanty lze přečíst z vlastností `WebGLRenderingContext` rozhraní, ke kterému se získá přístup z canvas elementu. Stačí tedy vytvořit tento element a přečíst z něj konstanty.

Odchylka od UTC

UTC[10] je koordinovaný světový čas, který definuje časová pásma pomocí odchylek od sebe samotného. Například čas v České republice oproti UTC je v letním období UTC+2:00 (o 2 hodiny více) a v zimním období UTC+1:00 (o hodinu více). V kontextu fingerprintingu je tato informace velmi identifikující v případě mezinárodní návštěvnosti webu. Pokud by se například jednalo o webovou stránku cílenou pouze pro Českou republiku, unikátnost odchylky od UTC by byla velmi nízká, protože drtivá většina uživatelů by sdílela jednu hodnotu.

Objekt `Date` má vlastnost `timeZoneOffset` vyjadřující odchylku UTC v minutách. Tuto hodnotu vrací funkce `getTimezoneOffset()` zavolaná nad `Date` objektem vytvořeným z aktuálního času a data prohlížeče. JavaScript získává aktuální čas a datum prohlížeče na základě zařízení, na kterém je prohlížeč spuštěn.

Do Not Track

Jedná se o nepovinnou hlavičku HTTP požadavku zvanou DNT, která webům oznamuje, zda prohlížeč má nebo nemá být sledován. Nabývá boolean hodnot `true` a `false`, přičemž vyjádření nesouhlasu být sledován reprezentuje hodnota `true`. V ohledu na fingerprinting se DNT hlavička může zdát jako ne příliš cenná, jelikož nabývá pouze dvou hodnot, ale při celkovém porovnávání fingerprintů může být rozhodujícím faktorem. Většina moderních prohlížečů svým uživatelům umožňuje si toto nastavení změnit přímo v prohlížeči. Ukázalo se však, že tuto politiku dodržuje jen zlomek webů světového dosahu (například reddit.com nebo pinterest.com) [53] a její použití za účelem zvýšení soukromí na internetu je proto takřka bezpředmětné.

DNT hodnotu lze získat z vlastnosti `window.navigator.doNotTrack`, jejíž návratový typ se ovšem napříč prohlížeči liší. U nejpoužívanějších prohlížečů má vlastnost návratový typ `Boolean`, `Number` nebo `String`. Hodnota je pak v databázi aplikace reprezentována ve formě `true` a `false`.

AdBlock

AdBlock je druh rozšíření prohlížeče s úlohou blokovat zobrazení reklam na internetu. S přibývajícím množstvím markentingu na internetu je v dnešní době jeho oblíbenost mezi uživateli čím dál větší. Existuje obrovské množství AdBlock rozšíření, ale pomocí JavaScriptu nelze získat jejich název. Nejedná se totiž o plugin, ale o rozšíření prohlížeče, a k těm JavaScript přístup nemá. Jak bylo řečeno u DNT hlavičky, pro fingerprinting je dostačující i to, že informace nabývá dvou hodnot. Proto stačí zohlednit, jestli je AdBlock vůbec v prohlížeči přítomen, nebo ne.

Jak už bylo řečeno, nativně detekovat AdBlock pomocí JavaScriptu nelze. Existuje však osvědčené řešení, s nímž toho lze docílit. AdBlock funguje na bázi srovnávání obsahu aktuálně navštíveného webu se seznamem filtrů a pravidel indikujících přítomnost reklamy. Pokud se na webu například vyskytuje soubor s názvem používaným pro reklamu, AdBlock tento soubor

[10] Anglicky Coordinated Universal Time.

skryje pomocí DOM a CSS. Pomocí přidání souboru nazvaným `ad_banner.js` dojde k vytvoření falešné reklamy, kterou se AdBlock pokusí zablokovat. Do souboru stačí vložit například kód přidávající neviditelný HTML element, který bude sloužit pro následnou detekci blokování reklam. Pokud AdBlock tento soubor zablokoval, kód v něm se nespustil, HTML element se nevytvořil a AdBlock je přítomen.

Ostatní použitá data

Zde následuje krátký popis ostatních fingerprinting dat, která jsou v aplikaci využita, ale postup k jejich získání není zajímavý nebo rozmanitost jejich hodnot příliš velká.

- **Platforma prohlížeče**: Jedná se o textový řetězec určující platformu, na které prohlížeč běží. Tuto vlastnost lze číst pomocí `navigator.platform` [54].
- **Data nabývající dvou hodnot**: Kromě detekce DNT hlavičky a AdBlock pluginu jsou v aplikaci použity další informace nabývající dvou hodnot. Všechny opět nesou hodnotu `true`, nebo `false`, a indikují přítomnost technologie v prohlížeči. Jmenovitě jde o cookies, Web Storage API, Indexed Database API, úložiště userData, plugin Flash a plugin Java.

4.2 Serverová část

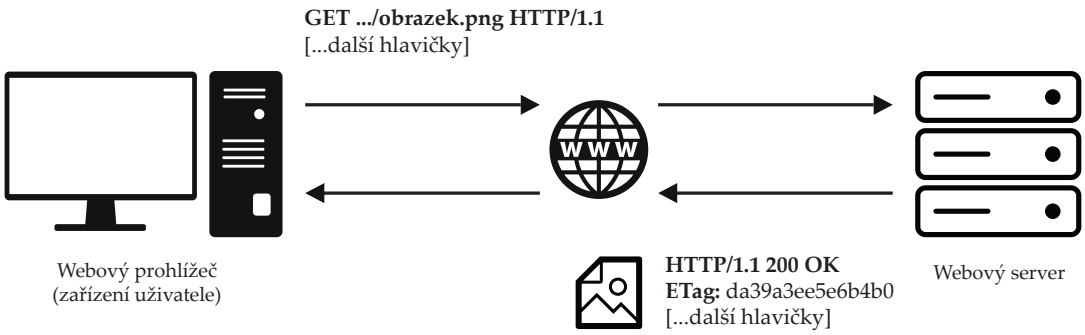
Serverová část aplikace je implementována ve frameworku ASP.NET. Využívá ETag hlavičku pro identifikaci uživatelů pomocí mezipaměti prohlížeče, a také obsahuje algoritmus pro otestování účelů fingerprintingu.

4.2.1 Mezipaměť prohlížeče s pomocí ETag

Velké množství důležitých informací již bylo řečeno v kapitole 3.3, proto bude implementace vysvětlena především na příkladu.

Situace v obrázku 6 znázorňuje první příchod uživatele na web. Popis událostí je následující:

- uživatel navštíví web poprvé
- JavaScript pomocí AJAX zažádá o neexistující obsah `obrazek.png`
- server obdrží požadavek a zkontroluje, jestli je přítomna hodnota hlavičky If-None-Match
- hodnota přítomna není, jelikož se jedná o uživatelovu první návštěvu
- server vygeneruje nový identifikátor a vloží záznam do databáze
- zároveň server přidá identifikátor jako hodnotu ETag hlavičky v HTTP odpovědi spolu s hlavičkou Cache-Control mající dobu expirace jednoho roku

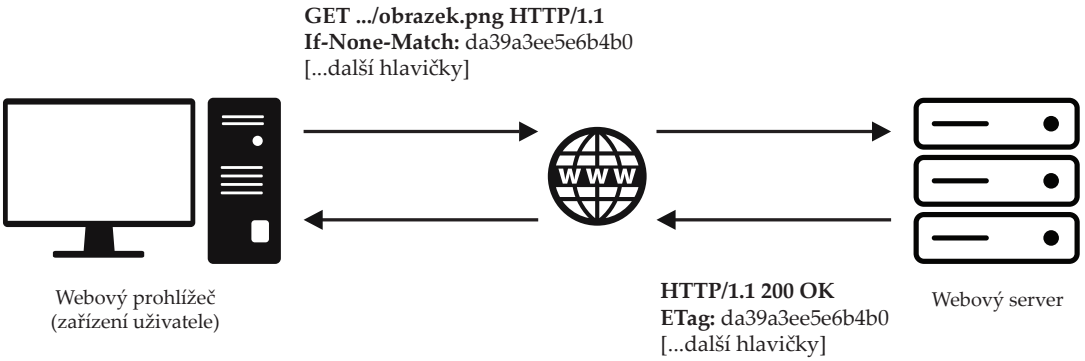


Obrázek 6: První návštěva webu (*Zdroj: vlastní zpracování)*

- uživatel obdrží obsah a prohlížeč jej na základě Cache-Control hlavičky uloží do cache spolu s ETagem

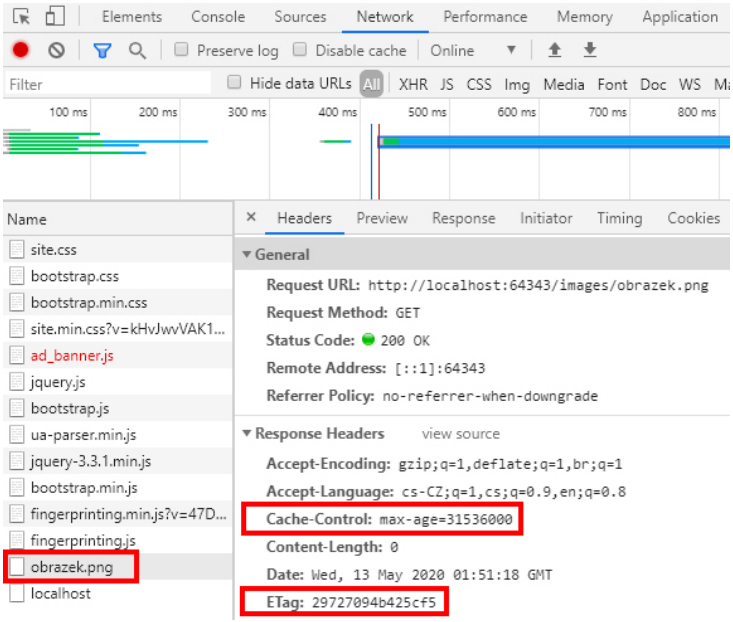
Obrázek 7 ukazuje uživatelovu opětovnou návštěvu. Popis událostí je následující:

- uživatel navštíví web podruhé
- JavaScript pomocí AJAX zažádá o neexistující obsah `obrazek.png`, ale tentokrát ve svém požadavku zašle také If-None-Match hlavičku s hodnotou ETagu z obsahu v mezipaměti
- server obdrží požadavek a zkontroluje, jestli je přítomna hodnota hlavičky If-None-Match
- hodnota přítomna je, takže ji server najde v databázi a aktualizuje záznam
- server přidá aktuální identifikátor zpět do ETag hlavičky a zašle HTTP odpověď s hlavičkou Cache-Control mající dobu expirace jednoho roku
- uživatel obdrží obsah a prohlížeč jej na základě Cache-Control hlavičky uloží do cache spolu s ETagem



Obrázek 7: Opětovná návštěva webu (*Zdroj: vlastní zpracování*)

Obrázek 8 ukazuje hlavičky HTTP odpovědi neexistujícího souboru `obrazek.png`.



Obrázek 8: Hlavičky HTTP požadavku v nástrojích pro vývojáře (Google Chrome).

4.2.2 Fingerprinting

Dvě jediné informace získávané pro účely fingerprintingu na straně serveru jsou hodnoty hlaviček Accept-Language a Accept-Encoding. Obě hlavičky se čtou z příchozího požadavku klienta na začátku návštěvy. Hlavička Accept-Encoding nebude popisována, jelikož byly její hodnoty u všech uživatelů totožné, a není tak pro práci zajímavá.

Accept-Language hlavička

Hlavička Accept-Language je zásílána jako součást HTTP požadavku ze strany klienta. Její smysl je sdělit serveru upřednostňované jazyky prohlížeče, aby například server mohl klientovi vrátit obsah webu v jeho preferovaném jazyku. Jednotlivý jazyk v hlavičce je definován jako tzv. *language tag* [49]. Sděluje název jazyka, popřípadě i jeho detailnější odnož, tzv. *region*. Dále také hodnotu kvality, která určuje prioritu vůči ostatním jazykům. Jazyky jsou podle hodnoty kvality seřazeny sestupně, od nejvíce preferovaného jazyku. Samotný nejvíce preferovaný jazyk je většinou bez dodatečné hodnoty kvality, což indikuje výchozí hodnotu 1. Následující řetězec `cs-CZ,cs;q=0.9,en;q=0.8` říká, že preferovaný language tag uživatele je `cs-CZ` s implicitní kvalitou 1, za ním následuje `cs` s kvalitou 0.9 a `en` s kvalitou 0.8. Tyto informace mohou otisk uživatele silně odlišit, především od uživatelů z jiné země.

Algoritmus na identifikaci otisků

Otisky uživatelů se s časem postupně mění. Může jít například o verzi prohlížeče, seznam nainstalovaných fontů a pluginů, výjimečně i některé hardwarové prvky zařízení. Hlavním účelem algoritmu bylo vyhnout se obyčejnému srovnávání jednotlivých informací otisku pouze na základě rovnosti. U některých informací je však žádoucí, aby byly srovnány přímo takto. Tyto informace budou nyní jmenovány a u některých z nich bude přidáno odůvodnění:

- název operačního systému
- typ zařízení
- název softwaru prohlížeče
- název platformy prohlížeče
- webGLVendor a webGLRenderer
- canvas fingerprint
- název prohlížeče (pouze u počítačů)
- název modelu zařízení (pouze u mobilních zařízení)
- název prodejce zařízení (pouze u mobilních zařízení)
- rozlišení zařízení (pouze u mobilních zařízení)
- informace o CPU zařízení (pouze u mobilních zařízení)

Srovnáním hodnot atributů vstupního uživatele se všemi existujícími uživateli v databázi se vyřadí všichni takoví, jejichž hodnoty atributů se nerovnají. Zbytek atributů otisku se srovnává pomocí algoritmu k-nejbližších sousedů, přesněji řečeno pomocí právě jednoho nejbližšího souseda. Pro hledání vzdáleností je použitá euklidovská metrika. Jelikož jsou jednotlivé atributy většinou velmi odlišeného charakteru, musely se jejich hodnoty vzdáleností přeškálovat tak, aby celkové srovnávání dávalo smysl. Data jsou navíc reprezentována většinou textovými řetězci a ne čísly.

4.2.3 Výsledná aplikace

Ve výsledné aplikaci je spojeno použití mezipaměti a fingerprintingu, což by v praxi mohlo představovat silnou kombinaci pro spolehlivější sledování uživatelů. Použity jsou také tří úložiště prohlížeče zmiňované dříve: localStorage, Indexed Database a cookies. Ty slouží pouze k vizuální kontrole originálního identifikátoru a ne k jeho oživování. Hodnoty jednotlivých úložišť se mění pouze v případě, kdy dané úložiště žádnou hodnotu neobsahuje. V takovém případě dojde k

uložení ETag hodnoty do tohoto úložiště. Ke kontrole slouží také jméno, které bylo uživateli přiřazeno (`assignedName`) a počet jeho návštěv (`visits`).

Pro případy, kdy nedojde k rozpoznání pomocí identifikátoru v ETag hlavičce, dojde teprve na fingerprinting. Algoritmus se pokusí uhodnout, o kterého uživatele z databáze by se mohlo jednat. Pokud nenajde dostatečnou shodu s žádným jiným záznamem v databázi, vloží aktuálního uživatele jako nového do databáze. Pokud dostatečnou shodu najde, předpokládá, že uživatel nalezený v databázi pouze smazal svůj ETag identifikátor. Postup je následující:

- Uživateli v databázi se nastaví atribut `newEtag`, čímž dojde k jeho označení za již neaktuálbího. Hodnota `newEtag` bude hodnota ETagu aktuálního uživatele.
- Aktuálnímu uživateli se před vložením do databáze nastaví jeho `assignedName` a `visits` podle hodnot uživatele v databázi. Zároveň se nastaví jeho hodnota `oldEtag` na hodnotu ETagu uživatele v databázi.
- Dojde ke vložení aktuálního uživatele do databáze.

Jeden původní uživatel může být takto přeidentifikován několikrát. Proto se vždy dává přednost záznamu, který má hodnotu `newEtag`, a zároveň nemá hodnotu `oldEtag`.

Může se samozřejmě stát, že algoritmus identifikuje uživatele špatně. Pokud původní uživatel navštíví web, díky hodnotě v `newEtag` se zjistí, že byl označen za neaktuálního, a proto dojde k opravení této chyby:

- Špatně identifikovanému uživateli se resetuje `oldEtag` atribut na prázdný řetězec.
- Jeho `visits` se odečtou od `visits` původního uživatele.
- Bude mu vygenerováno nové jméno a jeho záznam aktualizován.
- Původnímu uživateli se resetuje `newEtag` atribut na prázdný řetězec.
- Tímto se oba uživatelé vrátí do správného stavu, kdy ani jeden z nich nemá hodnoty atributů `oldEtag` a `newEtag`.

Složitější scénář nastává v případě, kdy došlo k několikanásobnému přeidentifikování jednoho uživatele. Situace je znázorněna na obrázku 1.

etag	assignedName	visits	oldEtag	newetag
375ede30105001c	Daniel Hulka	7		29727094b425cf5
29727094b425cf5	Daniel Hulka	13	375ede30105001c	b07fa1da2e2b5c4
b07fa1da2e2b5c4	Daniel Hulka	17	29727094b425cf5	

Tabulka 1: Znázornění přeidentifikování jednoho uživatele vícekrát.

5 Výsledky testování

Přesnost výsledků této kapitoly závisí na předpokladu, že každá zaznamenaná návštěva byla skutečně unikátní. Toto lze ověřit pouze s pomocí cookies, localStorage nebo Indexed Database. Pokud prohlížeč při návštěvě webu nemá identifikátor ani v jednom ze zmíněných úložišť, je brán jako nový záznam. Vygeneruje se mu unikátní identifikátor ve formě textového řetězce, který tvoří patnáct znaků, a zapíše se do zmíněných úložišť prohlížeče. Pokud tedy uživatel nevymazal svůj identifikátor ze všech použitých úložišť před provedením opětovné návštěvy, identifikátor bude nalezen a záznam v databázi bude pouze aktualizován. Při smazání identifikátoru pouze z některého úložiště se tento identifikátor "oživí" z kteréhokoli jiného úložiště. Jestliže existují uživatelé, kteří svůj identifikátor smazali ze všech úložišť, byla od nich data posbírána víckrát. Pro výsledky práce to znamená, že posbírané otisky mohly být doopravdy ještě více unikátní a fingerprinting metody o to účinější. Tuto možnou odchylku bohužel s jistotou nelze najít.

5.1 Mezipaměť prohlížeče s pomocí ETag

Testování probíhalo na 30 prohlížečích, z nichž 20 patřilo počítačům a 10 mobilním zařízením. Testovány byly prohlížeče: Google Chrome (na Windows), Google Chrome (pro Android), Mozilla Firefox, Opera, Microsoft Edge, Safari Mobile a Internet Explorer. Postup byl identický s popisem implementace této metody v kapitole 4.2.1. Identifikátor uživatele se při první návštěvě zároveň zapsal do tří úložišť: Indexed Database, localStorage a cookies. Tyto úložiště fungovaly pro zpětnou kontrolu. Při první návštěvě (všechny 3 úložiště byly prázdné) se záznam prohlížeče zapsal do databáze s hodnotou ETagu a hodnotami všech úložišť. V každém prohlížeči se web otevřel v jeden den (u některých byl web víckrát obnoven, u některých ne) a znovu se otevřel po třech dalších dnech. Pokud by se ETag některého prohlížeče nerovnal žádnému ETagu v databázi, tak by to znamenalo novou vygenerovanou hodnotu a záznam by se s ní zapsal do databáze spolu s původními hodnotami úložišť. Tímto způsobem by byly označeny záznamy s lišícími se hodnotami ETagu a úložištích. Toto se však u žádného záznamu nestalo, takže ve všech případech byla hodnota ETagu stejná a prohlížeč identifikován. Sledování uživatelů pomocí mezipaměti samozřejmě není možné v režimu anonymního prohlížení, protože při ukončení relace se smaže všechna tato cache z prohlížeče.

U všech prohlížečů lze smazat jejich cache přes nástroje pro vývojáře, ale to se od průměrného uživatele neočekává. Pokusy o smazání mezipaměti pro určitou webovou stránku dopadly pro různé prohlížeče odlišně:

- U prohlížečů Google Chrome, Mozilla Firefox a Opera se mezipaměť webu nesmazala při kombinaci kláves Ctrl + R, Ctrl + F5, ani Ctrl + Shift + R. U Microsoft Edge a Internet Explorer se to podařilo.

- Firefox mezipaměť úspěšně smazal při smazání dat daného webu skrz nastavení prohlížeče. Google Chrome a Opera ale neuspěli. Oba tyto prohlížeče jsou nádstavbou Chromium prohlížeče, a to je nejspíš důvod stejného chování.
- Kliknutím pravého tlačítka myši na tlačítko obnovení se v Chromu nabízí varianta *Vymazání mezipaměti a úplné opětovné načtení*, která cache konečně vymazala. V Opeře tato možnost není a jediná zbývající varianta pro průměrného uživatele je smazat cache v celém prohlížeči.
- Celkové smazání mezipaměti z prohlížeče funguje vždy.

Ačkoliv bylo využití webové mezipaměti úspěšné, její samostatné použití nepřináší tolik výhod. Oproti cookies je to rozhodně méně nápadný způsob identifikování, především pro průměrného uživatele, a šance na její smazání je také menší. Pokud se totiž uživatel rozhodne smazat pouze cookies, mezipaměť zůstane netknuta. Pokud chce ale uživatel smazat opravdu všechna data z prohlížeče, je velmi pravděpodobné, že zvolí také smazání cache a identifikátoru v ETag hlavičce se tak zbaví.

5.2 Fingerprinting

Aplikace a databáze byly hostovány na cloudové platformě Microsoft Azure. Během tří dnů bylo posbíráno 160 otisků prohlížečů pomocí šíření odkazu mezi přáteli, kolegy a známými, kteří byli s cílem aplikace dopředu obeznámeni. Od této skupiny lidí se proto očekává minimální, nebo dokonce žádná snaha o změnu svého otisku, a výsledky testování by proto v tomto ohledu neměly být zaujaté, ať už ve prospěch fingerprintingu, nebo naopak. Výsledky však silně závisí na velikosti získaných dat, jelikož 160 uživatelů dostatečně nesimuluje reálnou demografii internetu. Na objektivní zhodnocení úspěšnosti fingerprintingu v dnešní době by proto bylo potřeba udělat rozsáhlejší výzkum.

Informační entropie

Množství informací, které lze získat z jednotlivých částí otisků, může být popsáno v bitech jako neurčitost výsledku. Tato informační entropie H veličiny S o konečném počtu prvků $S\epsilon\left\{s_1, s_1, ..., s_n\right\}, n < \infty$ s pravděpodobnostní funkcí každého prvku $p(s_i)$ je definována jako:

$H(S) = -\sum_{i=1}^{n} P(s_i) log_2 P(s_i)$
kde $P(s_i) log_2 P(s_i) = 0 pokud P(s_i) = 0.$

	Počet bitů informace		
Posbíraná informace	**Celkem**	**Počítače**	**Mobilní zařízení**
User-Agent	6.02	3.93	6.01
Canvas (vylepšený)	5.83	4.53	5.10
Canvas	5.76	4.51	5.00
WebGL Renderer	5.41	5.21	3.82
Rozlišení	4.31	2.17	4.35
Accept-Language	4.15	3.95	3.27
Seznam fontů	3.55	4.21	1.20
WebGL Vendor	2.69	1.80	1.78
Platforma prohlížeče	2.34	1.17	1.51
Seznam pluginů	1.63	2.02	0
AdBlock	0.81	1.00	0.16
Do Not Track	0.47	0.72	0.09
Flash	0.31	0.52	0
Odchylka od UTC	0.21	0.20	0.19
Java	0.13	0.24	0
Accept-Encoding	0.06	0.10	0
Ostatní	0	0	0

Tabulka 2: Bity informace získané z jednotlivých metod

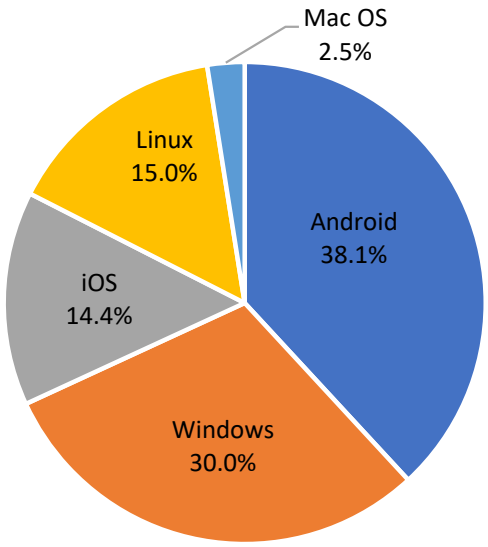
Podle tohoto vzorce musíme znát pravděpodobnostní funkci každé hodnoty dílčího prvku otisku, abychom mohli spočítat výslednou informační entropii tohoto prvku. Při aplikaci této teorie na identifikování fingerprintů pak platí, že každý 1 bit informace zmenší počet jiných vyhovujících fingerprintů o polovinu. Tabulka 2 popisuje získané bity informace pomocí jednotlivých metod. Z tabulky je zjevné, že úspěšnost fingerprinting metod se liší mezi počítači a mobilními zařízeními.

Nejzajímavější výsledky

Podíl mobilních zařízení v posbíraných datech byl 52,5 % a podíl počítačů byl 47,5 %. Toto je důležité brát na vědomí při vyhodnocování výsledků, protože se u těchto platforem předpokladá rozdíl v účinnosti fingerprintingu. Graf 9 ukazuje distribuci operačních systémů v posbíraných datech, graf 12 pak distribuci prohlížečů na počítačích a graf 13 distribuci prohlížečů na mobilních zařízeních. Bližším prozkoumáním byly vyvozeny následující výsledky:

- Celková unikátnost otisků byla 91,3 %. U počítačů bylo unikátních 94,7 % otisků, zatímco u mobilních zařízení to bylo 88,1 %, z čehož vyplývá, že mobilní zařízení odolávají fingerprintingu více než počítače.

- Žádný otisk se neopakoval více než dvakrát, tzn. pokud nebyl otisk unikátní, sdíleli ho právě 2 uživatelé. Tuto metriku (také známou jako *velikost anonymitní množiny*) znázorňuje graf 10, a to zvlášť pro počítače, mobilní zařízení a jednotlivé operační systémy. Z grafu lze vyčíst také počet unikátních otisků (velikost 1 anonymitní množiny) pro jmenované kategorie.



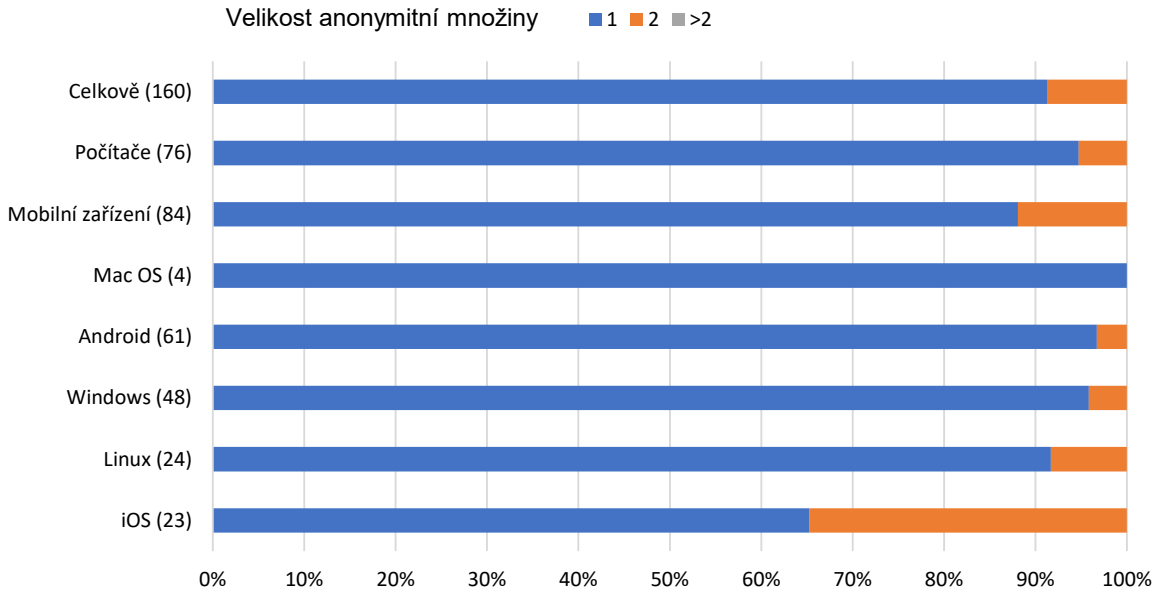
Obrázek 9: Distribuce operačních systémů

- Mezi mobilními zařízeními s operačním systémem Android a iOS jsou drastické rozdíly v počtu jejich unikátních otisků. Zatímco u iOS zařízení bylo unikátních pouze 65 % otisků, u Android zařízení to bylo 96.7 %. Můžeme tedy říct, že iOS zařízení jsou výrazně odolnější proti fingerprintingu než Android zařízení.

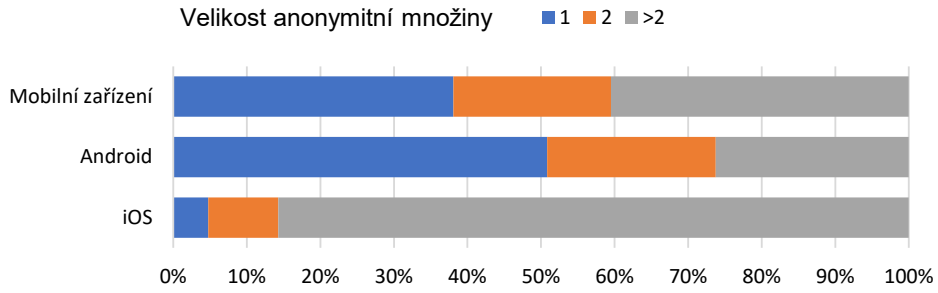
- Canvas fingerprinting (vylepšená) metoda měla unikátní hodnoty u celkových 36.3 % uživatelů. U mobilních zařízení to bylo 38 % a u počítačů 34 %. Mobilní zařízení však opět ukázaly obrovský rozdíl mezi operačními systémy. iOS zařízení měly pouze 4,4 % unikátních canvas otisků, zatímco Android zařízení 51 %. Z grafu 11 můžeme dále vyčíst, že 23 % Android zařízení sdílelo canvas otisk s jedním dalším zařízením, kdežto v případě iOS to bylo pouhých 9 %.

- Všechny mobilní zařízení měly stejnou hodnotu seznamu pluginů, tato informace přinesla 0 bitů informace.

- Nejúspěšnější fingerprinting metodou obou operačních systémů mobilních zařízení byl sběr User-Agent. Android měl 93,4 % unikátních hodnot a iOS 47,8 %.



Obrázek 10: Distribuce anonymitních skupin fingerprintů (tří velikostí) pro počítače, mobilní zařízení a jednotlivé operační systémy

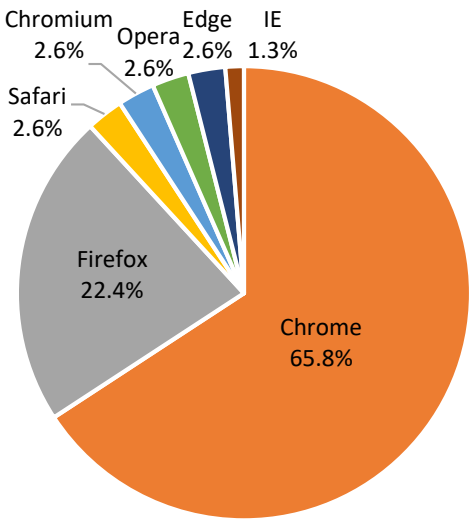


Obrázek 11: Distribuce anonymitních skupin canvas fingerprintů (tří velikostí) pro počítače, mobilní zařízení a jednotlivé operační systémy

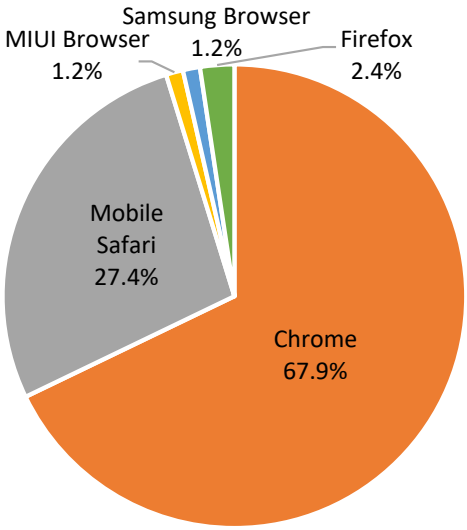
Rozdíly v úspěšnosti metod

U několika metod, jejichž celkový přínos v bitech informace patřil k těm nejvyšším, byl zjištěn poměrně odlišný výsledek mezi platformami nebo některými operačními systémy. Následuje rozebrání možných důvodů této rozdílnosti:

- **User-Agent**: Název a verze operačního systému jsou nejhodnotnějšími informacemi z User-Agent hlavičky. U Linuxu a distribucí z něj vycházejících (nezahrnuje Android) se ale podařilo verzi systému získat pouze u třetiny. Ve všech případech jí bylo Ubuntu. V ostatních případech totiž Linux přesnou verzi neposkytoval, o systému sděloval pouze



Obrázek 12: Distribuce prohlížečů na počítačích



Obrázek 13: Distribuce prohlížečů na mobilních zařízeních

generický text `X11; Linux x86_64`. Po bližším prozkoumání se potvrdilo, že toto nebyla náhoda a Linux systémy opravdu většinou neodkryjí přesnou verzi. Na první pohled to znamená lepší ochranu před fingerprintingem. Na druhou stranu se dá očekávat, že Linux systémy dostatečně identifikuje pouze jejich název, jelikož má mezi uživateli poměrně malé zastoupení.

- **Rozlišení zařízení**: Různorodost rozlišení displeje počítačů byla podle očekávání poměrně malá, kdy 67,1 % uživatelů mělo rozlišení `1920x1080`, které je dnes oblíbeným standardem. U mobilních zařízení byla křivka distribuce rozlišení mnohem více lineární, a tedy více vhodná pro fingerprinting. Celkově se rozlišení zařízení jeví jako spolehlivá metoda u obou platforem, i když u počítačů méně.

- **WebGL Renderer**: Počítače dosáhly 35,5 % unikátních hodnot a k tomu nemalých 34 % uživatelů sdílelo stejnou hodnotu právě s jedním dalším uživatelem. Zbylá distribuce velikosti skupin WebGL Rendereru byla téměř lineární. Díky těmto zjištěním a faktu, že průměrný uživatel mění svou grafickou kartu jen několikrát za život, je WebGL Renderer u počítačů nejvíce identifikující informací. Mobilní zařízení měly pouze 13 % unikátních hodnot a Androidu patřily všechny z nich. Zbytek hodnot Androidu byl rozprostřen zhruba lineárně. Na druhou stranu všechny iOS zařízení poskytly tutéž hodnotu, jelikož Apple sděluje pouze generický název jejich grafického čipu: `Apple GPU`. Pro iOS to opět vyřazuje použitelnost této metody fingerprintingu, která je jinak u zbytku zařízení silná.

- **Seznam fontů**: Velký rozdíl v bitech informace u počítačů a mobilních zařízení měl podobný výsledek jako seznam pluginů, kde byly všechny hodnoty totožné. IOS zařízení měly

stejný seznam fontů v 91,3 % případů a Android zařízení dokonce v 95,1 %. U Androidu pak existovaly jen 2 další jiné seznamy a u iOS pouze 1. Toto potvrdilo, že mezi uživateli není typické si do mobilních zařízení instalovat dodatečné fonty a tato metoda u nich tudíž nemá příliš smysl.

Obrázek 14: Vykreslený canvas fingerprinting obrázek na Windows 10 (fingerprintingjs2)

- **Canvas fingerprinting**: Jak již bylo zmíněno, implementovány byly dva způsoby canvas fingerprintingu. Hodnoty prvního řešení byly unikátní u 30,6 % uživatelů, zatímco u vylepšeného řešení od fingerprintingjs2 to bylo u 36,3 % uživatelů. Vylepšená metoda se tedy potvrdila jako schopnější zachytit menší nuance mezi zařízeními a prohlížeči. Dosahuje toho vykreslením dalších tvarů (kruhů) s vhodně zvoleným nastavením *canvas blending* a *canvas winding* možností. Canvas blending umožňuje míchání barev překrývajících se objektů díky zvolenému nastavení vlastnosti `globalCompositeOperation`, v tomto případě na hodnotu `multiply`. Canvas winding rozhoduje o výplni tvarů na základě zvoleného algoritmu, v tomto případě je použita možnost `evenodd`. Obrázek 14 reprezentuje vylepšenou verzi canvas fingerprintingu.

  Co se týče iOS zařízení a jejich velké imunity vůči canvas fingerprintingu, je na vině s největší pravděpodobností jednotný hardware (především grafický čip), který Apple do svých iPhone zařízení dává.

Metody s celkovým minimálním přínosem

Následující metody nepřinesly žádné nebo téměř žádné informace pro odlišení uživatelů, a můžeme o nich říct, že na výsledný fingerprint měly žádný nebo zanedbatelný vliv:

- **Odchylka od UTC**: Metoda posbírala pouze 4 různé hodnoty, což dáváme za důvod sbírání dat mezi cílovou skupinou jednoho časového pásma.
- **Accept-Encoding**: Hodnota této HTTP hlavičky se lišila pouze u jednoho uživatele.
- **Informace nabývající dvou hodnot**: Celkově bylo sbíráno 9 informací nabývajících dvou hodnot a žádnou unikátní hodnotu nepřineslo 5 z nich. Jmenovitě jimi jsou cookies, sessionStorage, localStorage, Indexed Database a úložiště UserData. Ve všech těchto případech byla hodnota `true`. Přítomnost pluginů Java a Flash přinesla jen velmi malou

odlišitelnost. O něco lépe na tom je přítomnost Do Not Track hlavičky a především rozšíření AdBlock. Přítomnost AdBlocku má 0.3 bitů informací, což už není tak zanedbatelná hodnota.

Úspěšnost identifikování algoritmem

Při testování algoritmu nastaly případy, kdy otisk uživatele nemohl být s jistotou identifikován, protože stejný otisk byl nalezen ještě nejméně u jednoho dalšího uživatele. V této situaci algoritmus vybral nejstarší z totožných otisků. Tento postup se může zdát jako ne příliš spolehlivý, což je samozřejmě do jisté míry pravda. Je ale nutné si uvědomit, že identifikace pomocí fingerprintingu je pouze experimentální metodou a v době psaní této práce neexistuje veřejně známý způsob, který by konzistentně zajišťoval stoprocentní přesnost. Úspěšnost algoritmu byla následující:

- Celkem 145 ze 160 uživatelů (90,6 %) bylo správně identifikováno.
- Celkem 15 ze 160 uživatelů (9,4 %) bylo identifikováno špatně, tedy jako někdo jiný. Pro statistiku úspěšnosti algoritmu tvoří tato hodnota chybu typu I, neboli falešně pozitivní (anglicky *false positive*).

Nastalo celkem 23 případů (14,4 %), kdy se algoritmus nemohl jednoznačně rozhodnout, protože se otisk jevil jako naprosto shodný s ještě nejméně jedním dalším otiskem. To znamená, že podle algoritmu bylo jednoznačně unikátních 85,6 % otisků. Tato čísla na první pohled nekorespondují s procentem celkových unikátních otisků v našich datech, které bylo 91,3 %. Zde hraje roli fakt, že algoritmus neporovnává přímo celé User-Agent řetězce, ale srovnává dílčí vyparsovaná data. U 60,7% otisků byla v User-Agent hlavičce nalezena stopa o otevření webu skrze nějakou aplikaci, tj. web byl otevřen ve formě WebView (Android) nebo UIWebView (iOS) a ne v samotném prohlížeči. Mohlo se však stále jednat o totožného uživatele. Pokud byl web otevřen například v aplikaci *Facebook Messenger*, do databáze se zapsal vyparsovaný název prohlížeče jako *Facebook* a číslování verze se také lišilo. Toto komplikuje srovnávání názvu a verze prohlížeče. Pro iOS platí, že při otevření webu v aplikaci se v User-Agent vůbec nevyskytuje informace o opravdovém názvu a verzi výchozího prohlížeče, který byl použit. U Android zařízení tyto informace poskytnuty jsou. Když tedy algoritmus u mobilních zařízení nenašel shodu názvu prohlížeče, pokusil se ještě vyparsovat Google Chrome verzi (nejčastější výchozí prohlížeč u Android zařízení), aby se mohl pokusit srovnat toto číslo namísto čísla poskytnutého aplikací. U iOS toto neprobíhalo.

U 10 uživatelů došlo ke shodnosti právě s jedním dalším. Pak následovaly tři skupiny shodných otisků: jedna o počtu 3 uživatelů, další o počtu 4 a poslední o počtu 6. Všechny 3 skupiny se skládaly pouze z iOS zařízení a kromě User-Agent řetězce byl mezi nimi rozdíl pouze v rozlišení, protože šlo o jiné modely. Toto zjištění pouze potvrzuje čísla z výsledků o sběru dat a můžeme o Applu říct, že jeho mobilní zařízení jsou příkladem odolávání proti fingerprintingu.

Bylo by zajímavé otestovat algoritmus na větším množství dat, aby se potvrdila jeho větší spolehlivost. Ideálně by data měla pocházet z jednoho webu se stálými návštěvníky a byla shromážděna po delší dobu (několik týdnů až měsíců). Díky tomu by mohla být lépe otestována schopnost algoritmu úspěšně identifikovat otisky uživatelů, které se za čas změnily. Změna otisku uživatele v čase je totiž největší problém fingerprintingu.

6 Obrana proti sledování

U většiny metod sledování je obrana proti nim poměrně přímočará. Stačí pravidelně mazat data uložená v prohlížeči (jejichž součástí jsou i identifikátory webů), nebo používat režim anonymního procházení, kde po ukončení dané relace nedojde k uložení žádných dat. Uživatelské pohodlí je pak ovšem velice sníženo, protože se každá návštěva webové stránky jeví jako první. Toto zahrnuje také větší odezvu, jelikož mezipaměť prohlížečů výrazně urychluje opětovné načítání webů. Dnešní prohlížeče nabízí možnost vypnutí cookies třetích stran, což výrazně sníží stopu uživatele při procházení webu a mohlo by to být dostačujícím opatřením pro zvýšení soukromí na internetu.

Mnohem složitější je obrana proti fingerprintingu:

- Nejjednodušší možností obrany proti fingerprintingu je používání co možná nejvíce používaného prohlížeče (Google Chrome) spolu s co nejmenší úpravou jeho nastavení.

- Vypnutím JavaScriptu v prohlížeči dojde k dostatečnému omezení poskytovaných informací a fingerprinting to téměř zněmožní. Toto s sebou ale nese obrovské omezení při procházení webu, protože mnoho webů je v dnešní době naprosto závislých na JavaScriptu a obsah webu se bez něj například ani nezobrazí. Webová stránka *gov.uk*, poskytující oficiální informace o veřejných záležitostech Spojeného království, ke konci roku 2013 zveřejnila článek zabývající se počtem svých návštěvníků, jejichž prohlížeč měl vypnutý JavaScript nebo jej vůbec nepodporoval [60]. Výsledkem byla statistika vycházející z více než půl milionu záznamů návštěv jejich webu. Tato statistika uvádí, že návštěvníků majících vypnutý JavaScript nebo používajících prohlížeč bez jeho podpory, bylo pouze 0,2 %. Statistiky webu *blockmetry.com* z roku 2016 ukazují podobná čísla [61]. JavaScript je navíc dlouhodobým a stále stoupajícím trendem při vývoji webu a nedá se tak předpokládát, že by v horizontu několika let přišel jeho ústup.

- nabízí v podobě používání některých rozšíření do prohlížečů. Asi nejznámějším takovým rozšířením je *Ghostery* [62], které zabraňují spouštění některých částí JavaScript kódu získávajícího informace z prohlížeče. Také existují rozšíření, které upravují User-Agent hlavičku tak, aby se prohlížeč (popřípadě i zařízení) na venek jevil jako jiný.

- Dosud nejspolehlivější známou ochranou před fingerprintingem je používání prohlížeče Tor. Tor je založen na speciální verzi prohlížeče Firefox a to Mozilla Firefox ESR. Jedním z jeho hlavních cílů je ochránit anonymitu uživatelů do co největší míry. Fingerprintingu odolává tak, že na jakémkoli zařízení maskuje informace do podoby totožného otisku. Toho se snaží docílit například tím, že má jednotnou User-Agent hlavičku (vypadající jako Windows zařízení), zakazuje různou HTML5 funkcionalitu, vypíná JavaScript, nastavuje velikost okna prohlížeče na jednu generickou hodnotu a mnoho dalších. [63]

V posledních letech začal boj proti fingerprintingu. V listopadu roku 2019 vydal Apple dokument, který mluví o tom, jak jejich prohlížeč Safari (na všech jejich platformách) zvyšuje soukromí svých uživatelů. Ohledně fingerprintingu Apple říká, že znemožňuje některé jeho metody a diverzitu poskytovaných informací se snaží redukovat na minimum. Toto lze potvrdit také na základě výsledků čtvrté kapitoly této bakalářské práce, kdy iOS zařízení předvedly velkou imunitu proti metodám fingerprintingu. Prohlížeč Mozilla Firefox nabízí možnost zapnutí funkcí *privacy.resistFingerprinting*[11] a *webgl.disabled*[12]. Firefox se těmito opatřeními snaží následovat prohlížeč Tor. Také blokuje všechny HTTP požadavky třetích stran, které vedou od společností známých pro sledování uživatelů pomocí fingerprintingu [66].

[11] Zamezuje odhalení některých nastavení prohlížeče uživatele, například časové pásmo nebo upřednostnění jazyků [65]

[12] Vypíná v prohlížeči podporu WebGL API.

7 Závěr

V teoretické části této práce byl nejprve vysvětlen účel cookies a popsán jejich vývoj. Dále se práce zabývala alternativními metodami sledování uživatelů na internetu bez použití cookies. Byl popsán princip jejich fungování, jejich výhody a úskalí.

V praktické části práce pak byly do vlastní aplikace implementovány dvě tyto metody, konkrétně fingerprinting a použití ETag HTTP hlavičky v mezipaměti prohlížeče. Výsledky úspěšnosti těchto metod byly zvlášť popsány v další kapitole. U metody s využitím ETag hlavičky se testovala odolnost mezipaměti u různých prohlížečů. Celkově se tato metoda jeví jako spolehlivá a může být vhodnou náhradou samotných cookies. V rámci metody fingerprintingu bylo posbíráno 160 otisků skutečných uživatelů a vyhodnocení jejich unikátnosti detailně rozebráno. Uživatelé měli unikátní otisk v 91,3 % případů, přičemž iOS zařízení se ukázaly jako mnohem hůře identifikovatelné než kterékoli jiné. Zjistilo se, že nejvíce unikátní informací u mobilních zařízení byla User-Agent hlavička HTTP požadavku s 93,4 % unikátních hodnot u Android zařízení a 47,8 % u iOS. Pomocí jednoduchého algoritmu na srovnávání otisků bylo správně identifikováno 90,6 % uživatelů a špatně 9,4 %. Tím, že se výsledky úspěšnosti fingerprintingu lišily u různých zařízení, tak fingerprinting nelze označit za spolehlivou metodu identifikace uživatelů, a proto je vhodné ji používat spíše jako doplňující alternativu k již zavedeným metodám. Výhodou je však to, že se drtivá většina uživatelů neumí proti tomuto způsobu sledování bránit.

Poslední část práce se zabývala možnými způsoby ochrany uživatelů před jednotlivými metodami sledování, ať už se jedná o opatření, které může uživatel podniknout sám, nebo o nástroje třetích stran, například rozšíření do webových prohlížečů nebo dokonce samotné webové prohlížeče, které slouží pro ochranu anonymity uživatelů.

Literatura

[1] FIELDING, R. a RESCHKE, J. RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing [online]. IETF Tools: June 2014 [cit. 6.12.2019]. Dostupné z: https://tools.ietf.org/html/rfc7230.

[2] Curl. Client Side State - HTTP Cookies [online]. Curl: ©1994 [cit. 10.1.2020]. Dostupné z: https://curl.haxx.se/rfc/cookie_spec.html.

[3] GVU Center. GVU's WWW User Surveys [online]. Georgia Institute of Technology. GVU Center, College of Computing: 29.06.2001 [cit. 26.12.2020]. Dostupné z: https://www.cc.gatech.edu/gvu/user_surveys/.

[4] Microsoft News Center. Microsoft Continues Commitment to Cross-Platform Solutions; Ships Microsoft Internet Explorer 2.0 for Windows 3.1 - Stories [online]. Microsoft: 1996 [cit. 20.2.2020]. Dostupné z: https://news.microsoft.com/1996/04/30/microsoft-continues-commitment-to- cross-platform-solutions-ships-microsoft-internet-explorer-2-0-for-windows-3-1/.

[5] KRISTOL, D. a MONTULLI, L. RFC 2109 - HTTP State Management Mechanism [online]. IETF Tools: February 1997 [cit. 3.2.2020. Dostupné z: https://tools.ietf.org/html/rfc2109.

[6] KRISTOL, D. a MONTULLI, L. RFC 2965 - HTTP State Management Mechanism [online]. IETF Tools: October 2001 [cit. 3.2.2020]. Dostupné z: https://tools.ietf.org/html/rfc2965.

[7] BARTH, A. RFC 6265 - HTTP State Management Mechanism [online]. IETF Tools: April 2011 [cit. 4.2.2020]. Dostupné z: https://tools.ietf.org/html/rfc6265.

[8] KRISTOL, D. HTTP Cookies: Standards, Privacy, and Politics [online]. Stevens Institute of Technology: 2001 [cit. 4.2.2020]. Dostupné z: https://www.cs.stevens.edu/nicolosi/classes/17fa-cs578/ref4-1.pdf.

[9] RAUSCHMAYER, A. Speaking JavaScript [online]. Sebastopol, CA. O'Reilly Media: 20.02.2014 [cit. 15.1.2020]. Dostupné z: http://speakingjs.com/es5/index.html.

[10] World Wide Web Consortium. W3C Document Object Model [online]. W3C: 2005 [cit. 26.2.2020]. Dostupné z: https://www.w3.org/DOM/.

[11] MDN Web Docs. HTTP cookies [online]. Mozilla and individual contributors: © 2005-2020 [cit. 3.3.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

[12] BRAY, T. RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format [online]. IETF Tools: December 2017 [cit. 3.3.2020]. Dostupné z: https://tools.ietf.org/html/rfc8259.

[13] W3Schools Online Web Tutorials. JavaScript Versions [online]. W3Schools: © 1999-2020 [cit. 02.04.2019]. Dostupné z: https://www.w3schools.com/js/js_versions.asp.

[14] World Wide Web Consortium. Same Origin Policy [online]. W3C: 01.12.2009 [cit. 4.3.2020]. Dostupné z: https://www.w3.org/Security/wiki/Same_Origin_Policy.

[15] Treasure Data Blog. Enterprise Customer Data Platform. What's 3rd Party Cookie, and how is it used to track users? [online]. Treasure Data Blog, Arm Limited: 2017 [cit. 6.3.2020]. Dostupné z: https://blog.treasuredata.com/blog/2017/02/16/whats-3rd-party-cookie-and-how-is-it-used-to-track-users/.

[16] MATUSZEWSKA, K. First-Party vs Third-Party Cookies: Why First-Party Is the Way to Go [online]. Piwik PRO: 2018 [cit. 6.3.2020]. Dostupné z: https://piwik.pro/blog/first-party-vs-third-party- cookies-why-first-party-is-the-way-to-go/.

[17] World Wide Web Consortium. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification [online]. W3C: 16.04.2002 [cit. 10.32020]. Dostupné z: https://www.w3.org/TR/P3P/.

[18] Microsoft Docs. P3P is no longer supported (Windows) [online]. Microsoft: 2016 [cit. 10.3. 2020]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie- developer/compatibility/mt146424(v=vs.85).

[19] KOPECKÁ, K. Cookies: technologické a legislativní aspekty [online]. Brno. Masarykova univerzita, Filozofická fakulta: © 2013 [cit. 29.1.2020]. Dostupné z: https://is.muni.cz/th/ymbke/Cookies_Kopecka.pdf.

[20] BUJLOW, T., et al. Web Tracking: Mechanisms, Implications, and Defenses [online]. Cornell University, Digital Library: 28.07.2015 [cit. 21.2. 2020]. Dostupné z: https://arxiv.org/pdf/1507.07872.pdf.

[21] W3Schools Online Web Tutorials. Window name Property [online]. W3Schools: © 1999-2020 [cit. 25. 2. 2019]. Dostupné z: https://www.w3schools.com/jsref/prop_win_name.asp.

[22] World Wide Web Consortium. HTML5 A vocabulary and associated APIs for HTML and XHTML [online]. W3C: 2014 [cit. 22.2.2020]. Dostupné z: https://www.w3.org/TR/2014/REC-html5- 20141028/.

[23] HTML5test. How well does your browser support HTML5? [online]. HTML5test: 2019 [navštíveno 21.04.2019]. Dostupné z: https://html5test.com/results/desktop.html.

[24] MDN Web Docs. Web Storage API [online]. Mozilla and individual contributors: © 2005-2020 [cit. 20.4.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API.

[25] W3Schools Online Web Tutorials. HTML5 Web Storage [online]. W3Schools: © 1999-2020 [cit. 28. 2. 2019]. Dostupné z: https://www.w3schools.com/HTML/html5_webstorage.asp.

[26] World Wide Web Consortium. Indexed Database API 2.0 [online]. W3C: 30.1.2018 [cit. 22.4.2020]. Dostupné z: https://www.w3.org/TR/IndexedDB-2/.

[27] MDN Web Docs. The structured clone algorithm [online]. Mozilla and individual contributors: © 2005-2020 [cit. 20.4.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Structured_clone_algorithm.

[28] Google Developers. Working with IndexedDB [online]. Google Developers: 2018 [cit. 21.4.2020]. Dostupné z: https://developers.google.com/web/ilt/pwa/working-with-indexeddb.

[29] SOLTANI, A., et al. Flash Cookies and Privacy [online]. Social Science Research Network (SSRN): 11.08.2009 [cit. 23.4.2020]. Dostupné z: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1446862.

[30] MCDONALD, A. and CRANOR, L. A Survey of the Use of Adobe Flash Local Shared Objects to Respawn HTTP Cookies [online]. CyLab Security and Privacy Institute: 31.01.2011 [cit. 13.2.2020]. Dostupné z: https://www.cylab.cmu.edu/_files/pdfs/tech_reports/CMUCyLab11001.pdf.

[31] W3Techs. Historical yearly trends in the usage of client- side programming languages for websites [online]. W3Techs: © 2009-2020 [cit. 10.04.2019]. Dostupné z: https://w3techs.com/technologies/history_overview/client_side_language/all/y.

[32] Adobe Blog. Flash & The Future of Interactive Content [online]. Adobe: 25.07.2017 [cit. 10.4.2020]. Dostupné z: https://theblog.adobe.com/adobe-flash-update/.

[33] CIMPANU, C. Google Chrome: Flash Usage Declines from 80% in 2014 to Under 8% Today [online]. Bleeping Computer: @ 2003 – 2020 [cit. 13.2.2020]. Dostupné z: https://www.bleepingcomputer.com/news/security/google-chrome-flash-usage-declines-from-80-percent-in-2014-to-under-8-percent-today/.

[34] Microsoft Docs. IsolatedStorageFile.IncreaseQuotaTo Method [online]. Microsoft: 2011 [cit. 18.1.2020]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/silverlight/dotnet- windows-silverlight/cc626384(v=vs.95).

[35] Microsoft Support. Search product lifecycle [online]. Microsoft: 2019 [cit. 20.04.2019]. Dostupné z: https://support.microsoft.com/en-us/lifecycle/search?alpha=Silverlight.

[36] MDN Web Docs. Etag [online]. Mozilla and individual contributors: © 2005-2020 [cit. 22.4.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/ETag.

[37] MDN Web Docs. If-None-Match [online]. Mozilla and individual contributors: © 2005-2020 [cit. 22.4.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match.

[38] MDN Web Docs. 304 Not Modified [online]. Mozilla and individual contributors: © 2005-2020 [cit. 22.4.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/304.

[39] FIELDING, R. a RESCHKE, J. RFC 7232 - Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests [online]. IETF Tools: June 2014 [cit. 12.12.2019. Dostupné z: https://tools.ietf.org/html/rfc7232.

[40] MDN Web Docs. Last Modified [online]. Mozilla and individual contributors: © 2005-2020 [cit. 02.04.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Last-Modified.

[41] MDN Web Docs. If-Modified-Since [online]. Mozilla and individual contributors: © 2005-2020 [cit. 02.04.2019]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-Modified-Since.

[42] ACAR, G., EUBANK, C., ENGLEHARDT, S., JUAREZ, M., NARAYANAN, A., DIAZ, C. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild [online]. ACM: 2014, strany 674–689 [cit. 12.12.2019]. Dostupné z: https://doi.org/10.1145/2660267.2660347

[43] ENGLEHARDT, S., NARAYANAN, A. Online Tracking: A 1-million-site Measurement and Analysis [online]. ACM: 2016, strany 1388–1401 [cit. 15.12.2019]. Dostupné z: https://doi.org/10.1145/2976749.2978313

[44] ECKERSLEY, P. How Unique Is Your Web Browser? [online]. Electronic Frontier Foundation: 2010 [cit. 20.12.2019]. Dostupné z: https://panopticlick.eff.org/static/browser-uniqueness.pdf.

[45] LAPERDRIX, P., RUDAMETKIN, W., BAUDRY, B. (2016). Beauty and the Beast: Diverting modern webbrowsers to build unique browser fingerprints [online]. 37th IEEE Symposium on Security and Privacy (S&P 2016), strany 878-894 [cit. 23.12.2019]. Dostupné z: https://hal.inria.fr/hal-01285470/file/beauty-sp16.pdf#cite.acar14.

[46] MOWERY, K., SHACHAM, H. Pixel Perfect : Fingerprinting Canvas in HTML 5 [online]. University of South: © 2012 [cit. 27.12.2016]. Dostupné z: https://hovav.net/ucsd/dist/canvas.pdf.

[47] SALMAN, F. UAParser.js – A JavaScript-based User-Agent string parser [online]. GitHub, Inc.: © 2020 [cit. 27.12.2019]. Dostupné z: https://github.com/faisalman/ua-parser-js/.

[48] HAAG, J., VASILYEV, V. Fingerprintjs2 – Modern & flexible browser fingerprint library v2 [online]. GitHub, Inc.: © 2020 [cit. 28.12.2019]. Dostupné z: https://github.com/Valve/fingerprintjs2/.

[49] World Wide Web Consortium. Q&A about BCP 47 - Tags for Identifying Languages [online]. W3C: 22.02.2011 [cit. 4.1.2020]. Dostupné z: https://www.w3.org/International/questions/qa-choosing-language-tags.

[50] Cover, T.; Thomas, J. Elements of Information Theory 2nd Edition. Wiley Series in Telecommunications and Signal Processing, WileyInterscience, 2006, ISBN 0471241954, 13–31 pp.

[51] MDN Web Docs . Screen [online]. Mozilla and individual contributors: © 2005-2020 [cit. 20.04.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Screen.

[52] MDN Web Docs. Window.devicePixelRatio [online]. Mozilla and individual contributors: © 2005-2020 [cit. 20.04.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Window/devicePixelRatio.

[53] All About DNT. Companies that have implemented Do Not Track [online]. FPF. All About Do Not Track [cit. 20.03.2019]. Dostupné z: https://allaboutdnt.com/companies/.

[54] MDN Web Docs. Navigatorid.Platform [online]. Mozilla and individual contributors: © 2005-2020 [cit. 25.03.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/NavigatorID/platform.

[55] Khronos. WebGL Overview [online]. The Khronos Group Inc.: © 2020 [cit. 01.04.2020]. Dostupné z: https://www.khronos.org/webgl/.

[56] MDN Web Docs. WEBGL_Debug_Renderer_Info [online]. Mozilla and individual contributors: © 2005-2020 [cit. 26.03.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WEBGL_debug_renderer_info.

[57] MDN Web Docs. NavigatorPlugins.plugins [online]. Mozilla and individual contributors: © 2005-2020 [cit. 05.04.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/NavigatorPlugins/plugins.

[58] NIKIFORAKIS, N., KAPRAVELOS, A., JOOSEN, W., KRUEGEL, C., PIESSENS, F., VIGNA, G. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting [online]. IEEE Symposium on Security and Privacy 2013, strany 541-555: 25.06.2013 [cit. 6.4.2020. Dostupné z: https://ieeexplore.ieee.org/document/6547132.

[59] World Wide Web Consortium. HTML Canvas 2D Context [online]. W3C: 19.11.2015 [cit. 6.4.2020]. Dostupné z: https://www.w3.org/TR/2dcontext/.

[60] HERLIHY, P. How many people are missing out on JavaScript enhancement? [online]. Government UK: 2013 [cit. 6.4.2020]. Dostupné z: https://gds.blog.gov.uk/2013/10/21/how-many-people-are-missing-out- on-javascript-enhancement/.

[61] Blockmetry. What percentage of browsers with javascript disabled? [online]. Blockmetry: © 2016-2019 [cit. 8.4.2020]. Dostupné z: https://blockmetry.com/blog/javascript-disabled.

[62] Ghostery browser extension [browser extension] [cit. 9.4.2020]. Informace o produktu dostupné z: https://www.ghostery.com/products/.

[63] LAPERDRIX, P. Browser Fingerprinting: An Introduction and the Challenges Ahead [online]. The Tor Project: 04.08.2019 [cit. 9.4.2020]. Dostupné z: https://blog.torproject.org/browser-fingerprinting-introduction-and-challenges-ahead

[64] Apple. Safari Privacy Overview [online]. Apple Inc.: November 2019 [cit. 12.4.2020]. Dostupné z: https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf.

[65] User Tritter. Security/Fingerprinting [online]. MozillaWiki: 28.10.2016 [cit. 21. 3. 2019]. Dostupné z: https://wiki.mozilla.org/Security/Fingerprinting.

[66] ENGLEHARDT, S. Firefox 72 blocks third-party fingerprinting resources [online]. Mozilla Security Blog: 07.01.2020 [cit. 12.4.2020]. Dostupné z: https://blog.mozilla.org/security/2020/01/07/firefox-72-fingerprinting/.

[67] KAMKAR, S. Evercookie - virtually irrevocable persistent cookies [online]. Samy's web : 11.10.2010 [cit. 10. 2. 2019]. Dostupné z: https://samy.pl/evercookie/.

[68] Microsoft Docs. userData Behavior [online]. Microsoft: 2013 [cit. 15.4.2020]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions//ms531424(v=vs.85).

[69] Microsoft Docs. Element behaviors and HTCs are no longer supported [online]. Microsoft: 2016 [cit. 26.4.2020]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/windows/internet- explorer/ie-developer/compatibility/hh801216(v=vs.85).